View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Diposit Digital de Documents de la UAB

Universitat Autònoma
de Barcelona

GENERACIÓ I AUTOMATITZACIÓ D'AJUDES ON-LINE

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Sistemes
realitzat per

Alejandro Álvarez Guirao

i dirigit per

Jordi Pons Aróztegui

Escola d'Enginyeria

Sabadell, juliol de 2011

El sotasignat, ***Jordi Pons Aróztegui***,

professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per en ***Alejandro Álvarez Guirao*** i per a que consti firma la present.

Sabadell, ***juliol*** de ***2011***.

Signat: ***Jordi Pons Aróztegui***

El sotasignat, ***Oscar Lechago Oller***,

de ***UNIT 4 Ibérica***,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per en ***Alejandro Álvarez Guirao*** i per a que consti firma la present.

Sabadell, ***juliol*** de ***2011***.

Signat: ***Oscar Lechago Oller***

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol: GENERACIÓ I AUTOMATITZACIÓ D'AJUDES ON-LINE

Autor: Alejandro Álvarez Guirao **Data:** juliol 2011

Tutors: Jordi Pons Aróztegui i Oscar Lechago Oller

Titulació: Enginyeria tècnica en informàtica de sistemes

Paraules Clau:

- Documentació, generació d'ajudes, automatització.
- Documentación, generación de ayudas, automatización.
- Documentation, helps generation, automation.

El projecte té la finalitat de crear una aplicació en la plataforma Java pel departament de Documentació de l'empresa UNIT4 Ibérica, per tal de poder crear i desenvolupar les ajudes on-line dels productes que tenen actualment en el mercat. S'ha desenvolupat en les oficines que té UNIT4 Ibérica a Barberà del Vallès.

El proyecto tienen la finalidad de crear una aplicación en la plataforma Java para el departamento de Documentación de la empresa UNIT4 Ibérica, para poder crear y desarrollar las ayudas on-line de los productos que tienen actualmente en el mercado. El proyecto ha sido desarrollado en las oficinas de UNIT4 tiene en Barberá del Vallés.

The project will aim to create an application in the Java plataform for the Department of Documentation of UNIT4 Ibérica company, to create and develop on-line help support of its products that are currently on the market. The project has been developed in the offices of UNIT4 in Barberá del Vallés.

ÍNDEX

INTRODUCCIÓ

CONVENI UAB I UNIT4 IBÈRICA

Aquest projecte s'ha desenvolupat dins el marc d'un conveni de col·laboració entre la Universitat Autònoma de Barcelona i l'empresa UNIT4 Ibérica, el qual té una durada de 560 hores, compreses entre el mes de novembre de 2010 i el mes de juliol de 2011, durant les quals l'alumne realitzarà un treball de desenvolupament de software. Gràcies a aquest conveni, l'alumne pot adquirir experiència laboral en una empresa de desenvolupament de software de gestió i de tecnologies de la informació, així com la seva estada i treball en l'empresa li serveix com a projecte de fi de carrera per a la universitat.

L'EMPRESA

UNIT 4 Ibérica és la filial espanyola del grup UNIT4, un dels primers fabricants internacionals de software de gestió empresarial (ERP i CRM per a pimes, RRHH, BI, CPM).

L'empresa no va començar sent UNIT4 Ibérica, sinó que té els seus orígens en CCS, l'antic Centre de Càlcul de Sabadell fundat el 1963, el qual va ser la primera empresa espanyola de software i serveis per a la gestió d'empreses. Al 2006 una companyia Holandesa anomenada UNIT4 Agresso va comprar el 100% de les accions de CCS, i l'empresa va passar a formar part de la filial espanyola Agresso Spain, amb el nom de CCS Agresso, competint amb els altres dos grans fabricants de ERP com són SAP i Microsoft. A principis de 2010 el conjunt d'empreses de UNIT4 Agresso va decidir que per temes de màrqueting, entre d'altres, el millor era unificar-les totes, agafant un nom en comú. Aquest és UNIT4.

UNIT4 Ibérica desenvolupa, comercialitza i dóna serveis al seu complet i avançat software de gestió (ERP i CRM) tant per a pimes com grans empreses, públiques i privades en dos centres de I+D+i a Espanya, on més de 100 investigadors certificats pel Ministeri d'Indústria i una plantilla especialitzada de 600 persones desenvolupen les seves solucions per aquest software. Alguns d'aquests softwares són UNIT4 ekon, UNIT4 Agresso Business World i Ready (autèntic ERP per a pimes petites), amb els quals és el fabricant líder del país amb una important cartera de clients tant en empreses, com en les administracions públiques i la sanitat, que confien en aquestes solucions. La companyia també disposa d'oficines, centres de desenvolupament i serveis a Portugal i Guinea Equatorial.

L'empresa, internament, es divideix en diferents àrees segons la funció que es realitza. El sector que es dedica al desenvolupament de software s'anomena fàbrica. Donada la naturalesa específica del projecte, aquest es situa dins de fàbrica, concretament en el grup karat de recerca i desenvolupament, que és el subgrup que s'encarrega del desenvolupament i el suport de l'eina karat, la qual és una eina desenvolupada per l'empresa, que facilita el desenvolupament de noves aplicacions, així com el manteniment de les ja existents. En aquest marc el departament de documentació, està dins de fàbrica i s'encarrega de generar totes les ajudes referents a l'aplicació així com documents corporatius.

DESCRIPCIÓ

UNIT4, com s'ha comentat en el punt anterior, es dedica a crear i personalitzar software a través de les seves eines de programació i gestió. Una part molt important és la documentació d'ajuda d'aquestes aplicacions, així com les parts personalitzades que es fan a diferents clients, ja que molts dels usuaris que fan servir les seves eines tenen pocs coneixements informàtics i necessiten suport. Avui dia, hi ha un departament de Documentació dins l'empresa dedicat a crear l'estructura i el contingut d'aquestes ajudes. Actualment, el procés que tenen de creació d'ajuda no és gaire eficient en termes d'aprofitament de temps, i ha obligat a que els documentalistes hagin hagut de rebre formació per obtenir certs coneixements informàtics per poder corregir errors del procés i del programari que utilitzen.

Un altre aspecte a tenir en compte és el canvi de plataforma de Windows a Java en el conjunt d'eines utilitzades per l'empresa. Això ha fet que les ajudes hagin sigut modificades internament (estructura interna dels arxius), i, per tant, actualment s'han d'utilitzar diferents aplicacions per tal de fer un sol procés, cosa que no és gaire eficient.

Els documentalistes, han de tenir cura de conceptes informàtics que teòricament no tindrien que ser responsabilitat seva, fent que no estiguin concentrats únicament en la seva feina que és documentar.

Per tot això es planteja la necessitat de revisar els procediments actuals de treball del departament de Documentació i crear una nova eina per tal de millorar el rendiment del departament i fer més fàcil la seva feina.

OBJECTIUS

A nivell personal els objectius que es pretén assolir són els següents:

- Adquirir experiència en el treball diari dins d'una gran empresa i poder aprendre noves metodologies de treball.
- Assolir bons nivells tècnics en programació orientada a objectes i en el llenguatge Java.
- Desenvolupament d'una eina a partir de zero.

Els objectius generals del projecte en l'àmbit de UNIT4 són els següents:

- Crear una aplicació que faciliti i optimitzi la feina al departament de Documentació
- Que es puguin reutilitzar els projectes antics de manera fàcil.
- Concentrar tots els processos de treball que tenen actualment i que fan amb diferent programari en un de sol.

CONTINGUT DE LA MEMÒRIA

A continuació descriurem el contingut de la memòria en els diferents capítols que té.

En el primer capítol, "Introducció", tenim una petita introducció al context de l'empresa i del projecte a desenvolupar.

En el segon capítol, "Què és una ajuda?", trobarem una explicació detallada sobre el que és una ajuda i de què està formada, per tal d'entendre una mica més com treballa el departament de documentació i la complexitat del projecte.

En el tercer capítol, "Estudi de viabilitat", trobarem una explicació sobre l'estudi de viabilitat que s'ha fet del projecte i on analitzem els diferents punts per veure la seva viabilitat.

En el quart capítol, "Planificació del projecte", veurem explicat la planificació que es va preveure abans de començar el projecte.

En el cinquè capítol, "Fase d'anàlisi", explicarem els requisits funcionals i no funcionals que es van determinar a partir d'un anàlisis de la situació i dels objectius a assolir.

En el sisè capítol, “Fase de disseny”, explicarem detalladament, com s’ha dissenyat i estructurat l’aplicació, així com les llibreries externes que s’han fet servir i les peculiaritats del disseny.

En el setè capítol, “Fase de codificació i proves” trobarem una explicació sobre com s’ha estructurat el codi del projecte, així com els estils que s’han seguit i les proves realitzades per veure el correcte funcionament de l’aplicació.

I per finalitzar s’han realitzat les conclusions del projecte tant a nivell personal, com a nivell de UNIT4. Veurem quins objectius s’han assolit, quins han quedat pendents i proposarem unes línees de millora i ampliació.

QUÈ ÉS UNA AJUDA?

PARTS D'UNA AJUDA

Normalment l'ajuda que nosaltres podem visualitzar en la majoria de programari del mercat del software, està formada per documents HTML i uns fitxers auxiliars que estableixen l'estructura jeràrquica d'aquests fitxers.

Aquests documents HTML són pàgines web, com les que podem trobar per Internet, que tenen el contingut desitjat de l'ajuda amb totes les seves característiques: imatges, hipervincles, taules, etc.

Aleshores, hi ha una sèrie de fitxers, transparents a l'usuari final, que fan que l'ajuda tingui una estructura i un sentit.

Degut a que en aquest àmbit que ens trobem dins l'empresa tractarem el format d'ajuda de Windows, i el nou format d'ajuda de Java, a continuació explicarem l'estructura de cadascun d'ells, ja que són diferents en ambdós casos.

AJUDA EN FORMAT WINDOWS

Arxius que el formen

Arxius HTML: Arxius que contenen el contingut de l'ajuda en format web.

Arxiu .HHC: Arxiu que conté la taula de continguts, és a dir, l'estructura jeràrquica de l'ajuda. Ens indica la jerarquia que segueixen els diferents arxius HTML, quin és el seu pare, quins estan dins d'una carpeta, etc. Aquest arxiu està també en format HTML, com mostra aquest exemple:

```
<BODY>
<UL>
   <LI><OBJECT type="text/sitemap">
      <param name="Name" value="UNIT4 ekon. Módulo SAT">
      <param name="Local" value="UNIT4_ekon__M_dulo_SAT.htm">
       </OBJECT>
   <UL>
      <LI><OBJECT type="text/sitemap">
         <param name="Name" value="Cómo funciona la gestión del parque material">
         <param name="Local" value="C_mo_funciona_la_gesti_n_del_parque_material.htm">
          </OBJECT>
```

```
      <LI><OBJECT type="text/sitemap">
        <param name="Name" value="Cómo funciona la gestión de contratos">
        <param name="Local" value="C_mo_funciona_la_gesti_n_de_contratos.htm">
         </OBJECT>
   </UL>
</UL>
</BODY>
```

Arxiu .ALI: Aquest arxiu conté la relació entre el nom de l'arxiu físic de l'arxiu HTML, i el nom que figurarà en el visor d'ajuda (IDH), com mostra aquest exemple:

```
IDH Com_fer_una_ajuda_? = ajuda.html
```

En aquest exemple podem observar que l'arxiu 'ajuda.html' es mostrarà com a "Com fer una ajuda ?" en el visor de l'ajuda.

Arxiu .H: Aquest arxiu conté una relació entre els IDH dels arxius (noms que tenen a l'aplicació) i un número (ID). Això és perquè en algunes aplicacions, com és el nostre cas, ens interessa poder vincular un arxiu concret de l'ajuda, amb una part concreta de l'aplicació. Per exemple, si nosaltres estem mirant l'ajuda "com crear un usuari nou" estaria molt bé poder anar a l'aplicació en aquest punt concret, i a l'inversa, si estem en l'aplicació i volem anar a l'ajuda, que anéssim directament a aquest apartat. Així doncs aquest ID que pot anar lligat l'arxiu HTML (no és obligatori), serveix per vincular l'ajuda amb punts concrets de l'aplicació.

```
#define IDH_UNIT4 1001
```

Aquí podem veure com el IDH_UNIT4 està vinculat al ID 1001. Això farà que tinguem un vincle per exemple amb un formulari que tingui la ID 1001 dins de l'aplicació.

Com s'executa

En format Windows, un cop s'ha compilat una ajuda, es crea un arxiu .CHM. Aquest arxiu és un executable que crida al visor d'ajudes de Windows, i especifica on trobem els arxius esmentats anteriorment per tal de poder mostrar l'ajuda correctament.

AJUDA EN FORMAT JAVA

Arxius que el formen

Els arxius HTML són els mateixos que en el format de Windows. Però hi ha arxius diferents com mostrem a continuació.

Arxiu .TOC: Aquest arxiu seria comparable a l'arxiu .HHC del format Windows. És el que conté l'estructura jeràrquica de l'ajuda, i està en format XML, com mostrem a continuació.

```
<tocitem  text="UNIT4 ekon. Módulo SAT" target="1001">
  <tocitem text="Cómo funciona la gestión del parque material"
        target="C_mo_funciona_la_gesti_n_del_parque_material.htm" />
```

Arxiu map.XML: Aquest arxiu vincula l'arxiu físic HTML amb un 'target' (nom abstracte que agafa dins de l'ajuda). Aquest 'target', en cas de tenir el document de l'ajuda un ID, serà el ID com es mostra en l'exemple de sota. Aquest arxiu seria l'equivalent a l'arxiu .ali i .h del format Windows.

```
<mapID target="1001" url="UNIT4_ekon__M_dulo_SAT.htm" />
```

Arxiu HelpSet.hs: Aquest arxiu conté la informació bàsica pel visor d'ajuda de Java. Conté la informació dels arxius que fa servir per mostrar-les, així com la configuració del visor.

Com s'executa

En el nostre cas, l'execució de l'ajuda Java, la fem a través de codi. No tenim un executable. El que fem és fer servir la biblioteca d'Oracle Java Help, que és la oficial, i a partir d'aquesta biblioteca cridem a diferents funcions per tal de crear el visor a partir de l'arxiu HelpSet.hs que hem esmentat abans. El codi és el següent:

```
URL helpSetURL = new URL("file:///" + paramProyecto.getPathHTML() + "HelpSet.hs");

Help.setHelpEncoding("UTF-8");
Book libro = (Book) new HelpSet(helpSetURL);
Help ayuda = new Help();
ayuda.addBook(libro);
```

```
ayuda.setVisible(true);
ayuda.showNavigatorWindow(libro);
```

Com podem observar, el primer que fem és crear la URL on està l'arxiu HelpSet.hs. A continuació definim el tipus de codificació que farà servir i ,per últim, creem el visor.

ESTUDI DE VIABILITAT

SITUACIÓ ACTUAL

Context

El departament de Documentació, com hem esmentat abans, és l'encarregat de crear i mantenir les ajudes dels productes de l'empresa. Per fer aquesta tasca actualment realitzen el següent procés:

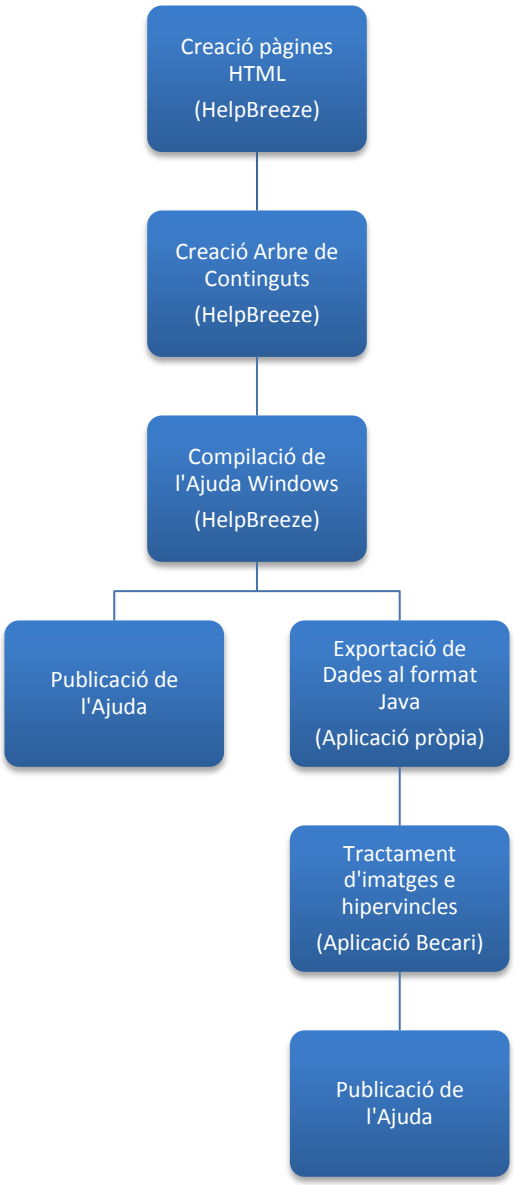

Figura 1 Procés actual

Primer de tot, els documentalistes creen les pàgines HTML amb el contingut de les ajudes, a la vegada que creen l'arbre de continguts. Un cop han fet això, i després de

posar les id's dels formularis corresponents, compilen l'ajuda en format Windows amb la mateixa eina. Aleshores, aquesta ajuda es pot publicar en aquest format, però si es vol publicar també en format Java és necessari fer una exportació de les dades. El problema és que aquestes han de ser tractades pels hipervincles i les imatges, per tal de que es vegin correctament en l'ajuda i per finalitzar la publicació de l'ajuda.

Com es pot observar el procés és poc eficient i comporta errors durant la compilació, degut a que el HelpBreeze[1] ja no ofereix suport, i en cas de voler noves eines de treball s'havien de pagar. També cal esmentar que l'editor HTML que porta integrat genera errors, i fa que els documentalistes hagin tingut que adquirir coneixements de llenguatge HTML per tal de poder modificar les pàgines web mitjançant el codi, tasca que no correspon a un documentalista.

Aleshores, un cop creat el contingut de l'ajuda es procedeix a la seva compilació en format Windows, i a la seva posterior publicació.

Paral·lelament, amb un programa desenvolupat per un programador de UNIT4, s'agafen els arxius que es generen i fa un tractament per crear els arxius necessaris per poder mostrar l'ajuda en Java. Però això no és suficient ja que s'han de tractar posteriorment les imatges i hipervincles, per tal de fer les rutes relatives i que es mostrin correctament. Això es fa mitjançant una altra aplicació creada per un becari en anys anteriors. Un cop fet aquest procés es pot mostrar l'ajuda en format Java.

Conclusions

Les conclusions que podem extreure són que és un procés poc eficient de cara a l'empresa i poc còmode a l'hora treballar de cara a l'empleat degut a la utilització de diferents aplicacions per fer una sola tasca. Un altre problema d'aquest procés és el manteniment de les ajudes, ja que cada cop que s'ha de fer una petita modificació, s'ha de realitzar el procés sencer, sent costós en temps, sobretot en projectes de gran dimensió.

Aleshores, crear una aplicació que concentri el procés de creació d'ajuda en format Java, facilitaria molt la feina dels documentalistes, així com es veuria augmentat el seu rendiment.

[1] Programa que fan servir per l'edició de l'ajuda en format Windows.

OBJECTIUS I ANÀLISIS DE REQUISITS

Tipologia i paraules claus

El projecte està enquadrat dintre de la categoria de desenvolupament de sistemes de software.

Les paraules claus per definir el projecte són: ajuda, automatització i documentació.

Descripció i priorització d'objectius

Després de diverses reunions amb el departament de documentació, de valorar les seves opinions i d'avaluar el procés de treball que tenen actualment, hem pogut extreure els següents objectius:

1. Automatitzar el procés de Documentació de les ajudes.
2. Millorar la qualitat de la feina dels Documentalistes.
3. Implantació d'un Editor HTML dins de l'aplicació.
4. Possibilitar la importació de projectes en format Windows a Java.
5. Facilitar la modificació d'ajudes ja existents.
6. Implementar una vista prèvia de l'ajuda per poder fer un seguiment sense necessitar una compilació.
7. Implementar un gestor d'errors per tal d'assegurar el correcte funcionament de l'ajuda (Log).
8. Implementació de fulls d'estil dins de l'aplicació.

	Crític	Prioritari	Secundari
O1	x		
O2		x	
O3		x	
O4	x		
O5	x		
O6		x	
O7		x	
O8			x

Taula 1 Objectius i prioritats

Parts interessades

A continuació presentem les parts interessades del projecte.

Nom	Descripció	Responsabilitat
Dept. Documentació UNIT4	Departament de l'empresa UNIT4 dedicat a la part documental de l'empresa.	Departament encarregat de crear i gestionar les ajudes dels diferents productes de l'empresa.

Taula 2 Parts interessades

Equip de projecte

En la taula següent es mostra com queda confeccionat l'equip del projecte així com una petita explicació de les tasques de cadascú.

Nom	Descripció	Responsabilitat
Jordi Pons Aróztegui	Tutor del Projecte (DP)	Supervisa la feina de l'alumne durant el projecte.
Oscar Lechago Oller	Cap de Projecte UNIT4 (CP)	Defineix, gestiona i controla el projecte.
Alejandro Álvarez Guirao	Analista (A)	Desenvolupa l'estudi de viabilitat i la planificació. Anàlisis de l'aplicació: arquitectura, metodologia, especificació, estàndards... Participa en el disseny i validació.
Josep Miquel Garcia	Analista (A)	Assessorament en l'àmbit d'Anàlisis de l'aplicació i assessorament tècnic al llarg del projecte.
Alejandro Álvarez Guirao	Programador (P)	Desenvolupa l'aplicació d'acord amb l'anàlisi i la planificació prevista. Fa la implantació del projecte.
Alejandro Álvarez Guirao	Dissenyador (D)	Dissenya i desenvolupa l'aspecte de l'aplicació d'acord amb l'anàlisi i normes d'usabilitat establertes.
Alejandro Álvarez Guirao	Tècnic de proves (TP)	Disseny de les proves i realització del procés de control de qualitat. Participa en el procés de proves.

Dept. Documentació	Tècnic de proves (TP)	Realització de les proves i participació en el control de qualitat.

Taula 3 Equip de projecte

Restriccions del sistema

A continuació exposem els diferents punts referents a les restriccions que tenim:

- L'aplicació ha de mantenir la imatge corporativa, utilitzant les icones de l'empresa així com els colors corporatius.
- S'ha de desenvolupar en un entorn Java per mantenir coherència amb les eines desenvolupades per l'empresa.
- L'aplicació ha d'adaptar-se a l'estructura informàtica tenint en compte el material informàtic que disposa el departament de Documentació.
- El projecte ha d'estar finalitzat abans del 30 de juny de 2011.

ALTERNATIVES

Al ser un projecte tancat dins de l'empresa UNIT4 no s'ha valorat la possibilitat d'alternatives per part de l'alumne. Ja que el projecte ja està definit amb uns objectius finals concrets. Per exemple no s'ha valorat el fet de modificar l'aplicació que fan servir, ja que és de codi tancat i té moltes funcionalitats que no són necessàries. També per política de reducció de costos s'ha desestimat l'adquisició d'un software per tal de fer les ajudes ja que solen ser d'un cost elevat i més si tenim en comptes que l'ús ha de ser corporatiu.

CONCLUSIONS

En conclusió podem afirmar que el desenvolupament d'aquesta aplicació suposarà una gran millora dintre del procés de treball de tot el departament de Documentació de l'empresa Unit4, degut a que faran en un sol procés tots els passos del seu flux de treball amb una considerable millora del temps empleat en realitzar la documentació, així com una millora en la comoditat a l'hora de realitzar-lo. A més, el programa estarà desenvolupat a mida a partir de les seves necessitats i indicacions, per tal de millorar el seu treball diari el màxim possible. I no només això, sinó que aquesta aplicació permetrà als clients de Unit4 poder modificar les ajudes en els productes fets a mida i així com als seus consultors.

PLANIFICACIÓ DEL PROJECTE

RECURSOS DEL PROJECTE

Aquí es presenta tot el referent a la planificació del projecte, així com quin material i quins recursos es faran servir.

Recursos del projecte

En aquest apartat podem observar els recursos econòmics, agafant els paràmetres genèrics aproximats del cost fictici suposat, encara que per aquest projecte l'alumne percebrà una quantitat real de 2.352€. El cost fictici calculat arrel de la planificació i dels paràmetres de la figura 5, el cost teòric del projecte seria de 48.880€.

Recursos Humans	Valoració
Cap de Projecte	100€/h
Analista	50€/h
Programador	30€/h
Tècnic proves	20€/h

Taula 4 Recursos econòmics teòrics

Calendari de recursos

Els recursos humans que s'utilitzaran durant el projecte són els següents:

- Cap de projecte: Iniciació, planificació, generació de documents, tancament i defensa del projecte. Punts de control.
- Analista: Anàlisi i disseny, implantació i punts de control d'anàlisi, disseny i desenvolupament.
- Programador: disseny, desenvolupament i test.
- Tècnic en proves: fases de test.

Els recursos materials s'utilitzaran principalment en les fases de desenvolupament, test i implantació.

CALENDARI DEL PROJECTE

El projecte es desenvoluparà de novembre de 2010 fins al juny de 2011, amb una dedicació aproximada de 20 hores setmanals amb un total d'hores dedicades al projecte d'aproximadament unes 560 hores.

Data de començament: 8 de novembre de 2010.

Data de finalització: 9 de juny de 2011.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Generació i automatització de ajudes on-line.**	**139 días**	**lun 29/11/10**	**jue 09/06/11**		
2	**Inici del projecte: Matriculació**	1 día	lun 29/11/10	lun 29/11/10		Programador
3	**Curs de Formació**	10 días	mar 30/11/10	lun 13/12/10	2	Programador
4	**Planificació**	**11 días**	**mar 30/11/10**	**mar 14/12/10**		
5	Estudi de viabilitat	1,5 días	mar 30/11/10	mié 01/12/10	2	Cap de Projecte;Analist
6	Aprovació Estudi Viabilitat	1 día	mar 14/12/10	mar 14/12/10	3	Cap de Projecte
7	Plà del Projecte	3 días	mié 01/12/10	lun 06/12/10	5	Cap de Projecte;Analist
8	Aprovació Plà de Projecte	1 día	lun 06/12/10	mar 07/12/10	7	Cap de Projecte
9	**Anàlisi de l'aplicació**	**10 días**	**mié 15/12/10**	**mar 28/12/10**		
10	Reunió Dept. Documentació (usuaris)	1 día	mié 15/12/10	mié 15/12/10	4	Analista
11	Anàlisi de requisits (cassos d'ús)	1 día	jue 16/12/10	jue 16/12/10	10	Cap de Projecte;Analist
12	Anàlisi de dades (fitxers)	3 días	vie 17/12/10	mar 21/12/10	11	Analista;Cap de Project
13	Anàlisi de la seguretat i de la legalitat	1 día	mié 22/12/10	mié 22/12/10	12	Analista
14	Documentació del Anàlisi	3 días	jue 23/12/10	lun 27/12/10	13	Analista
15	Aprovació de l'Anàlisi	1 día	mar 28/12/10	mar 28/12/10	14	Cap de Projecte
16	**Disseny de l'Aplicació**	**29 días**	**mié 29/12/10**	**lun 07/02/11**	**15**	
17	Disseny modular de l'Aplicació	20 días	mié 29/12/10	mar 25/01/11		Analista[50%];Program
18	Disseny de l'interfície gràfica	10 días	mié 29/12/10	mar 11/01/11		Analista[50%];Program
19	Disseny dels tests de proves	5 días	mié 26/01/11	mar 01/02/11	17	Cap de Projecte[50%];Pr
20	Documentació del disseny	3 días	mié 02/02/11	vie 04/02/11	19	Analista
21	Aprovació del Disseny	1 día	lun 07/02/11	lun 07/02/11	20	Cap de Projecte
22	**Desenvolupament de l'Aplicació**	**61,5 días**	**mar 08/02/11**	**mié 04/05/11**	**16**	
23	Preparació del entorn de desenvolupame	1 día	mar 08/02/11	mar 08/02/11		Programador
24	Configuració de les llibreries	0,5 días	mié 09/02/11	mié 09/02/11	23	Programador
25	Desenvolupament de les clases	60 días	mié 09/02/11	mié 04/05/11	24	Programador[50%]
26	Desenvolupament de l'interfície gràfica	20 días	mié 09/02/11	mié 09/03/11	24	Programador[50%]
27	**Test i proves**	**17 días**	**mié 04/05/11**	**vie 27/05/11**	**22**	
28	Proves unitàries	5 días	mié 04/05/11	mié 11/05/11		Tècnic en proves
29	Proves d'integració	2 días	mié 11/05/11	vie 13/05/11	28	Tècnic en proves
30	Proves d'estress	2 días	vie 13/05/11	mar 17/05/11	29	Tècnic en proves
31	Documentació de les proves	2 días	mar 17/05/11	jue 19/05/11	30	Tècnic en proves
32	Aprovació de les proves i resultats	1 día	jue 19/05/11	vie 20/05/11	31	Cap de Projecte
33	Correció d'errors	5 días	vie 20/05/11	vie 27/05/11	32	Programador
34	**Implantació**	**2,5 días**	**vie 27/05/11**	**mar 31/05/11**	**27**	
35	Instal·lació	0,5 días	vie 27/05/11	vie 27/05/11		Programador
36	Formació d'usuaris	2 días	lun 30/05/11	mar 31/05/11	35	Analista
37	**Generació de la Documentació**	**5 días**	**mié 01/06/11**	**mar 07/06/11**	**34**	
38	**Tancament del Projecte**	**1 día**	**mié 08/06/11**	**mié 08/06/11**	**37**	
39	**Defensa del Projecte**	**1 día**	**jue 09/06/11**	**jue 09/06/11**	**38**	

Figura 2 Diagrama de tasques

Aquest calendari pot variar degut a que està planificat per fer una jornada de 4 hores diàries, encara que això pot variar i fer una jornada més àmplia i reduir els dies. Aquí es pot observar amb el diagrama de Gantt la planificació prevista. També una de les coses que han fet variar ha sigut les noves propostes i requisits des del departament de documentació de funcionalitats noves de l'aplicació. Això ha provocat que dintre

de la fase de codificació, hagin hagut parts de disseny noves, per tal de implementar i inserir aquestes noves funcionalitats dintre del nostre disseny de la manera més fàcil i sense modificar gaire l'estructura interna de l'aplicació.

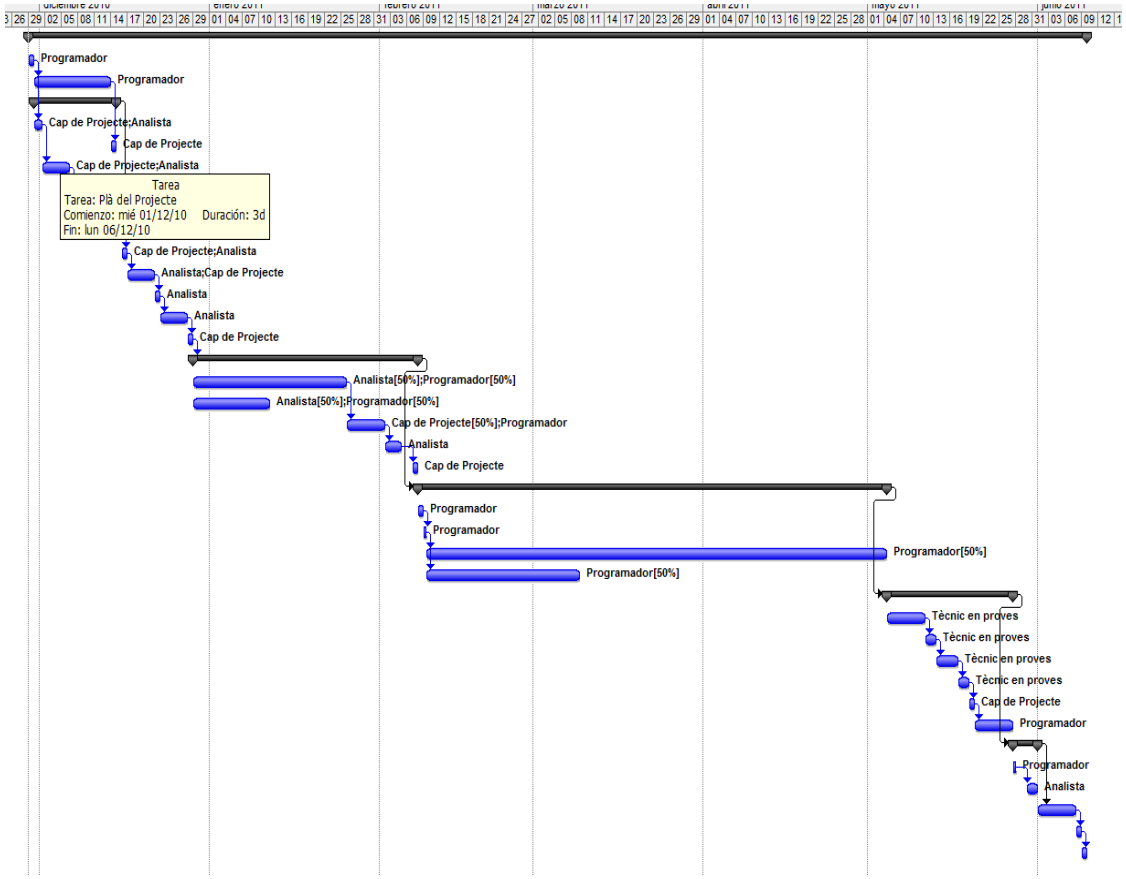

Figura 3 Diagrama de Gantt

Dependències

Totes les fases del projecte es desenvoluparan utilitzant un model lineal. Per tant, cada fase no es comença fins que no s'ha completat la fase anterior.

En la fase de desenvolupament, s'ha decidit utilitzar un model modular. L'aplicació s'ha dividit en diferents funcions (mòduls), i fins que no s'hagi acabat el mòdul actual i hagi passat la fase de proves modular, no s'avançarà al següent mòdul.

La fase de generació dels Documents serà al final degut a que inclourà els documents elaborats durant el desenvolupament del projecte, així com documents d'ajuda pel departament de Documentació.

AVALUACIÓ DE RISCOS

En aquest apartat podem observar els diferents tipus de riscos que es poden trobar al llarg del projecte amb les seves conseqüències i probabilitats. Com es pot veure els riscos més alts són la planificació optimista, sobretot per la falta de experiència en aquest tipus de projectes i problemes amb els components externs, en el nostre cas l'editor que fem servir, que pot no complir les necessitats.

Llista de riscos

Risc	Conseqüència
R1. Planificació temporal optimista	No s'acaba a temps, increment dels recursos
R2. Canvi de requisits	Endarreriment del producte.
R3. Fase de test incorrecta	Poca qualitat, deficiències operatives.
R4. Problemes amb components externs	Pèrdua de temps, deficiències operatives, no finalització del projecte.
R5. Falta de coneixements	Pèrdua de temps en formació, endarreriment del projecte.

Taula 5 Llista de riscos

Catalogació de riscos

	Probabilitat	Impacte
R1	Alta	Crític
R2	Mitja	Crític
R3	Mitja	Crític
R4	Mitja-Alta	Catastròfic
R5	Baixa	Crític

Taula 6 Catalogació de riscos

Pla de contingència

	Solució
R1	Retallar funcionalitats si no són crítiques. Augmentar hores de treball.

R2	Si són crítiques: retallar funcionalitats no crítiques. Augmentar hores de treball. Modificar la planificació.
R3	Dissenyar un test amb antelació, realitzar tests automàtics.
R4	Buscar un component alternatiu. Buscar una forma alternativa d'implementar la solució.
R5	Més hores de treball, retard del projecte.

Taula 7 Pla de contingència

RESUM I ANÀLISI COST-BENEFICI

Encara que segons els paràmetres generals, el cost tindria que ser de 48.880€, realment degut a que és un projecte dins d'una empresa com a becari, el cost és d'uns 2.352€ aproximadament en conceptes de desplaçament i de dietes.

L'amortització del projecte a nivell econòmic serà en un termini mig ja que l'aplicació està pensada pel Dept. de Documentació i per millorar la seva qualitat de treball diària. Aquesta farà que el temps invertit en una tasca es vegi reduïda considerablement. A més a més, la posterior correcció d'errors degut a males compilacions del programa ja que no serà necessària.

També podem dir que està previst vendre-la a clients per a que puguin generar la seva pròpia ajuda pels seus productes personalitzats. Amb el que es recuperarien els diners invertits d'una forma més ràpida, fins i tot obtenint beneficis econòmics.

FASE D'ANÀLISI

PERFILS D'USUARI

L'aplicació serà utilitzada per un sol tipus d'usuari, els documentalistes. Ells tenen control total sobre tots els projectes d'ajuda i poden modificar tots els seus paràmetres. Per tant, que no és necessari fer un control d'accés o de nivells de seguretat sobre els usuaris.

REQUISITS FUNCIONALS

RF1. Creació i manteniment de projectes d'ajuda

Els usuaris de l'aplicació han de poder crear un projecte d'ajuda des de zero. Ells han de poder crear una estructura jeràrquica de documents HTML, així com la seva edició i poder modificar l'estructura en qualsevol moment de forma fàcil i senzilla. Això implica poder crear nous tòpics o pàgines d'ajuda en qualsevol part de la estructura de l'ajuda, poder eliminar els tòpics clicant una sola opció, canviar el nom dels tòpics o dels arxius físics en qualsevol moment sense tenir que fer cap operació amb arxius, i poder modificar el contingut de les pàgines d'ajuda dins de la pròpia aplicació.

RF2. Importació de projectes antics

Els usuaris han de poder importar al nou format Java els projectes antics que hagin estat creats amb les eines antigues, de manera que no hagin de fer servir cap aplicació o procés entremig. Ells seleccionen l'origen del projecte antic, i l'origen on volem posar el nou projecte, i automàticament s'ha d'importar, creant els nous arxius de configuració i preparat per poder-ho modificar o ampliar de manera immediata, aplicant les noves directrius de codificació del projecte.

RF3. Funció de vista prèvia

Els usuaris han de poder visualitzar l'ajuda en qualsevol moment del procés de creació sense tenir que fer cap procés de compilació, per tal d'ajudar als documentalistes a veure com va evolucionant el seu treball sense tenir que esperar a tindre'l acabat i així poder veure el resultat actual per tal de poder fer modificacions, o poder veure el camí que estan seguint, ja que fins ara, cada cop que volien veure l'ajuda, tenien que compilar, un procés lent i que a vegades comportava errors, a més, de que mentre s'està fent aquest procés els documentalistes no poden estar modificant l'ajuda, cosa que en el nostre projecte si que volem permetre i veure els canvis en temps real.

RF4. Implantació d'un Editor HTML

L'aplicació ha d'integrar un editor HTML per tal de poder modificar dins de la mateixa aplicació els documents HTML de l'ajuda. L'editor serà un component extern ja que la programació de zero és massa costosa pel temps del projecte.

Aquest editor ha de tenir algunes funcions bàsiques que es fan servir molt com les llistes, tan numèriques com normal, la inserció d'imatges, poder fer servir una fulla d'estils, poder afegir capçaleres, afegir hipervincles, així com que sigui un editor WYSIWYG què és l'acrònim de "What You See Is What You Get" o el que és el mateix, "el que tu veus, és el que obtens" i és caracteritza perquè pots treballar a nivell de codi HTML i a nivell de resultat final, o fins i tot tenir la finestra amb les dos possibilitats a la vegada, encara que la opció de editar amb codi volem que no s'hagi de fer servir.

RF5. Controlador de plantilles

L'usuari ha de poder gestionar mitjançant un assistent les diferents plantilles que s'utilitzaran al llarg del procés de creació d'ajuda, podent afegir, eliminar o modificar les plantilles des de l'aplicació i que es faran servir a l'hora de crear nous tòpics, ja que la majoria dels tòpics segueixen una estructura bàsica depenent del tipus d'ajuda que sigui, per tal de facilitar i accelerar el màxim possible l'edició de les ajudes.

RF6. Generació de LOG

L'aplicació ha de generar un log intern per tal de poder fer un seguiment en cas d'errors per tal de poder localitzar-los i solucionar-los de forma ràpida i eficient.

REQUISITS NO FUNCIONALS

Restriccions i objectius de disseny

RDO1. Estàndards UNIT4

L'aplicació ha de seguir la imatge corporativa de l'empresa i ha de tenir una interfície 'user friendly', és a dir, que sigui senzilla i intuïtiva per l'usuari. Els icones han de estar dissenyats en funció de l'estètica de karat, l'eina que desenvolupa l'empresa Unit4. Degut a que és una empresa internacional amb presència amb tot el món, l'aplicació serà desenvolupada en anglès per tal de que sigui accessible per tota la corporació.

RDO2. Limitacions de hardware

El software a implementar ha de poder rendir de forma fluida en els equips del departament. Aquests tenen les característiques següents: processador de doble nucli, 2GB de memòria RAM, disc dur compartit en xarxa (espai suficient i ampliable). Respecte al sistema operatiu casi totes les màquines funcionen amb Microsoft Windows en la versió de XP o Seven, encara que aquest detall no és rellevant per nosaltres degut a que al córrer sobre la màquina virtual de Java, aquesta suporta tant Windows (en totes les versions), MacOs i Linux, amb el que no hi ha cap problema de restricció en la nostra aplicació en aquest àmbit.

RDO3. El software ha de ser ampliable

L'aplicació ha de tenir un codi clar i ben documentat per tal de que es puguin fer futures ampliacions o modificacions degut a possibles canvis en l'estructura de la creació de les ajudes. Per això es crearà un JavaDoc[2] i un document explicant el funcionament de cada classe i característica important del codi a nivell tècnic per tal de facilitar la feina dels tècnics.

[2] JavaDoc és una utilitat de Oracle per tal de generar la documentació en format HTML a partir dels comentaris inserits en el codi Java de la nostra aplicació.

FASE DE DISSENY

INTRODUCCIÓ

En aquest apartat tractarem els punts referents al tema del disseny i l'estructura interna de l'aplicació. Degut a que la nova eina de UNIT4 està programada sota la plataforma Java, es va decidir des del primer moment fer l'aplicació en aquesta plataforma, ja que no és depenent de cap sistema operatiu. També ens centrarem en les solucions que hem adoptat al llarg del disseny i la creació de l'aplicació per solucionar els diferents problemes que s'han anat presentant.

Durant aquesta fase s'ha rebut l'ajuda i l'assessorament de Josep Miquel Garcia, programador i analista de l'empresa Unit4 amb més de 10 anys d'experiència en aquest camp.

TECNOLOGIA DE DESENVOLUPAMENT

Java

Com hem comentat anteriorment Java serà el llenguatge que utilitzarem per crear la nostra aplicació. Java és un llenguatge de programació dissenyat al 1990 per James Gosling amb altres companys de Sun Microsystems a partir de C++. Des del seu naixement va ser pensat com un llenguatge orientat a objectes. Entre el 13 de novembre de 2006 i el maig del 2007 Sun va alliberar parts de Java com a programari lliure de codi obert amb llicència GPL. És un dels llenguatges de programació més utilitzats, i s'utilitza tant per aplicacions web com per aplicacions d'escriptori.

El Java és un llenguatge interpretat i, per tant, pot semblar lent en comparació amb altres llenguatges, però ofereix un índex de reutilització de codi molt elevat, sent possible trobar moltes llibreries lliures de *Java*. És un llenguatge flexible i potent tot i la facilitat amb la que es programa i dels resultats que ofereix. Un dels trets que el caracteritza i que el fa una eina molt valorada a l'hora de desenvolupar aplicacions distribuïdes, és el fet que és un llenguatge multiplataforma.

Generalment els programes de Java es compilen en un bytecode (fitxer .class) que pot córrer en una Màquina Virtual Java. Sun Microsystems disposa de tres implementacions diferents de *Java*: J2SE per a aplicacions d'escriptori; J2EE per a aplicacions distribuïdes i J2ME per a plataformes amb recursos més reduïts com ara

mòbils o PDAs. Per a cada una de les tres implementacions és possible descarregar el JRE *(entorn d'execució Java)* per a executar aplicacions o el SDK *(Eines per al desenvolupament d'aplicacions)* per a programar aplicacions en *Java*, aquest últim també inclou el JRE.

Un programa desenvolupat en *Java* no necessita compilar-se de nou per a poder executar-se en qualsevol de les plataformes que disposi d'una versió instal·lada de JRE prou actualitzada per al programa.

Algunes de les característiques de Java que fa que sigui una bona elecció pel nostre projecte són les següents:

Senzill: Java s'ha creat per a que sigui un llenguatge senzill amb una sintaxi elegant. Únicament consta de tres tipus de dades primàries, eliminant els punters i l'herència múltiple. Això fa que l'aprenentatge sigui més fàcil i ràpid.

Orientat a objectes: Java segueix els paradigmes de la programació orientada a objectes, ja que la programació amb Java es centralitza en la manipulació, creació i construcció d'objectes.

Robust: Java és un llenguatge robust i fiable, s'ha escrit pensant en poder verificar errors i està molt tipificat. Això és un detall molt important quan l'aplicació es farà servir en una multinacional amb un sistema informàtic complex, ja que garanteix una seguretat que avui dia és necessària.

Arquitectura neutral i portabilitat: Java és independent de la plataforma final on s'executarà el programa. Això fa que no s'hagin de tenir diverses versions per a cada sistema operatiu diferent, cosa que redueix costos de realització així com fa una millor optimització del codi.

Alt rendiment: Els compiladors Java han anat millorant les seves prestacions. Els nous compiladors coneguts com JIT permeten un rendiment molt semblant als llenguatges convencionals compilats.

Concurrent: Java permet l'execució de múltiples fils d'execució, o diverses tasques de forma simultània. Això és útil quan l'aplicació, com és el nostre cas, ha de fer tasques o processos de compilació mentre l'usuari fa unes altres tasques.

XML

Com hem esmentat en apartats anteriors, la majoria de fitxers que formen l'ajuda en format Java estan en format XML, per això a continuació explicarem de forma breu com és aquest llenguatge.

XML és un llenguatge de marques extensibles (eXtensible Markup Language). És un metallenguatge extensible d'etiquetes desenvolupat pel World Wide Web Consortium (w3c). És una simplificació i adaptació de l'experimentat SGML, i permet definir la gramàtica de llenguatges específics. Per tant XML, no és realment un llenguatge en particular, sinó una manera de definir llenguatges per a diferents necessitats. Alguns dels llenguatges que empren XML per a la seva definició són XHTML, SVG i MathML. XML no ha nascut només per a la seva aplicació a Internet, sinó que es proposa com a un estàndard per a l'intercanvi d'informació estructurada entre diferents plataformes. Es pot utilitzar per a bases de dades, editors de text, fulls de càlcul i per moltes altres aplicacions diverses. XML és una tecnologia relativament senzilla que té al seu voltant altres que la complementen i la fan notablement més extensa, a més de proporcionar-li unes possibilitats molt més grans. A l'actualitat té un paper molt important, ja que permet la compatibilitat entre sistemes, possibilitant la compartició d'informació d'una manera segura, fiable i fàcil.

La tecnologia XML busca donar solució al problema d'expressar informació estructurada de la manera més abstracta i reutilitzable possible. Per informació estructurada entenem que es compon de parts ben definides, i que aquelles parts es componen d'altres parts. S'aconsegueix un arbre amb trossos d'informació.

Una etiqueta consisteix en una marca feta al document, que senyala una porció d'aquest com un element, un tros d'informació amb un sentit clar i definit. Les etiquetes tenen la forma <nom>, on nom és el nom de l'element senyalat.

A continuació hi ha un exemple per a entendre l'estructura d'un document XML que fem servir per guardar part de la informació de l'aplicació:

```
<?xml version="1.0" encoding="UTF-8"?>
<conf>
      <altura text="1024" />
      <anchura text="1024" />
      <proj1 text="MIERCOLES" target="C:\workspace\MIERCOLES\" />
      <proj2 text="noau" target="C:\workspace\noau\" />
      <proj3 text="MIERCOLES" target="C:\workspace\MIERCOLES\" />
      <workspace text="C:\workspace\" />
      <plantillas text="c:\workspace\plantillas\" />
</conf>
```

Per manipular aquests arxius farem servir la llibreria externa JDOM de la qual en parlarem més endavant.

HTML

Com hem mostrat anteriorment totes les pàgines d'ajuda són documents HTML, així que ara tractarem aquest llenguatge per veure les seves particularitats.

HTML (acrònim d'*Hyper Text Markup Language*, en català, "llenguatge de marcat d'hipertext"), és un llenguatge de marcat que deriva de l'SGML dissenyat per estructurar textos i relacionar-los en forma d'hipertext. Gràcies a Internet i als navegadors web, s'ha convertit en un dels formats més populars que existeixen per a la construcció de documents per a la web.

Les etiquetes bàsiques d'HTML, d'obligada presència en tot document són:

- **<!DOCTYPE>:** És l'etiqueta que permet definir el tipus de document HTML que s'utilitzarà. Existeixen tres tipus bàsics: l'estricte (Strict), el transaccional (Transitional) i el de marcs (Frameset).
- **<html>:** És l'etiqueta arrel de qualsevol document HTML.
- **<head>:** Defineix la capçalera del document HTML. Permet declarar metainformació del document que no es mostra directament en el navegador. Aquesta informació és d'especial rellevància pels indexadors i cercadors automàtics.
- **<body>:** Defineix el cos del document. Aquesta és la part del document HTML que es mostra en el navegador.

- Dintre de la capçalera **<HEAD>** hi podem trobar:
  - **<title>:** Permet definir el títol de la pàgina.
  - **<meta>:** Permet definir meta informacions del document tals com l'autor, la data de realització, la codificació del document (UTF, ISO, etc.), les paraules clau i la descripció del mateix
  - **<LINK>:** Permet definir metadades complementàries a les del meta tals com el document anterior, el següent, el capítol al qual pertany el document, la pàgina glossari, etc.
- Dintre del cos **<BODY>** hi podem trobar:
  - **<a>:** Etiqueta àncora. Crea un enllaç a un altre document o a una altra zona del mateix, segons els atributs.

- **<h1>, <h2>,… <h6>:** capçaleres o títols del document, acostumen a distingir-se per mida.
- ** :** Divisió estructural de la pàgina.
- **<p>:** Paràgraf.
- **
:** Salt de línia.
- **<table>:** Indica el començament d'una taula, després s'haurà de definir les files amb **<tr>** i les cel·les dintre de les files amb **<td>.**
- **<ul>:** Llista desordenada (sense numerar). Els ítems es defineixen amb **<li>.**
- **<ol>:** Llista ordenada (numerat). Els ítems es defineixen amb **<li>.**

Excepte unes poques etiquetes, la majoria requereixen ser tancades escrivint la mateixa etiqueta precedida d'una barra "/". Exemple: **`<html>`**…**`</html>`**.

INTERFÍCIE GRÀFICA

En aquest apartat parlarem de quin ha sigut el procés per crear la interfície gràfica i què s'ha tingut en compte a l'hora de decidir-ho.

El primer que es va fer va ser mirar els programes destinats a crear ajudes per veure les diferents maneres d'estructurar els components en la interfície i quins posaven dins la interfície o no. Els programes que es van observar van ser: *Helpbreeze*, *HelpMaker, HelpCompilerWorkshop, WinCHM 3.3,* entre d'altres. Després d'investigar la seva interfície i provar-los, vam extreure diverses conclusions:

- **Barra d'eines:** l'aplicació hauria de tenir una barra d'eines amb icones grans i clares amb les funcions més utilitzades pels documentalistes. En la majoria d'aplicacions aquesta barra està situada en la part superior, encara que alguna aplicació la té a la banda esquerra.
- **Editor HTML:** totes les aplicacions en tenen un integrat, cosa que facilita molt la feina dels documentalistes i fa que la seva feina sigui més àgil i ràpida. En el nostre cas volem integrar un, el inconvenient que ha de ser un component extern ja que crear un des de 0, és una feina molt gran que no es pot incloure dintre dels termes d'aquest projecte.
- **Arbre de continguts:** totes les aplicacions l'incorporen, a més, són interactius. Es pot modificar l'arbre, cosa molt còmode pels documentalistes ja que és una de les tasques més habituals que fan, afegir documents d'ajuda o modificar-los dintre de l'arbre de continguts.

- **Propietats del projecte/tòpic:** alguns dels programes en la mateixa interfície mostren les propietats tant del projecte com del tòpic que tenen actiu en aquell moment. Més endavant quan parlem del components de l'aplicació explicarem perquè no es va incloure en la interfície principal.
- **ID del tòpic:** Aquí hi ha varietat entre els diferents programes. N'hi ha que aquest ID té un camp en la interfície principal, altres que has d'accedir a una finestra a través d'un menú o d'altres que has d'accedir a través de pestanyes dins de la interfície principal.

Després d'exposar tots aquest termes al departament de documentació i fer diferents reunions, es va fer un primer esborrany de l'aplicació. En la següent figura es pot observar el disseny del esborrany un cop finalitzat amb la diferenciació de les seves parts. Com es pot observar se li don molta importància al editor HTML i al arbre de continguts ja que són les principals eines que faran servir. Així com també un petit panell per poder escriure notes del projecte com a recordatoris.

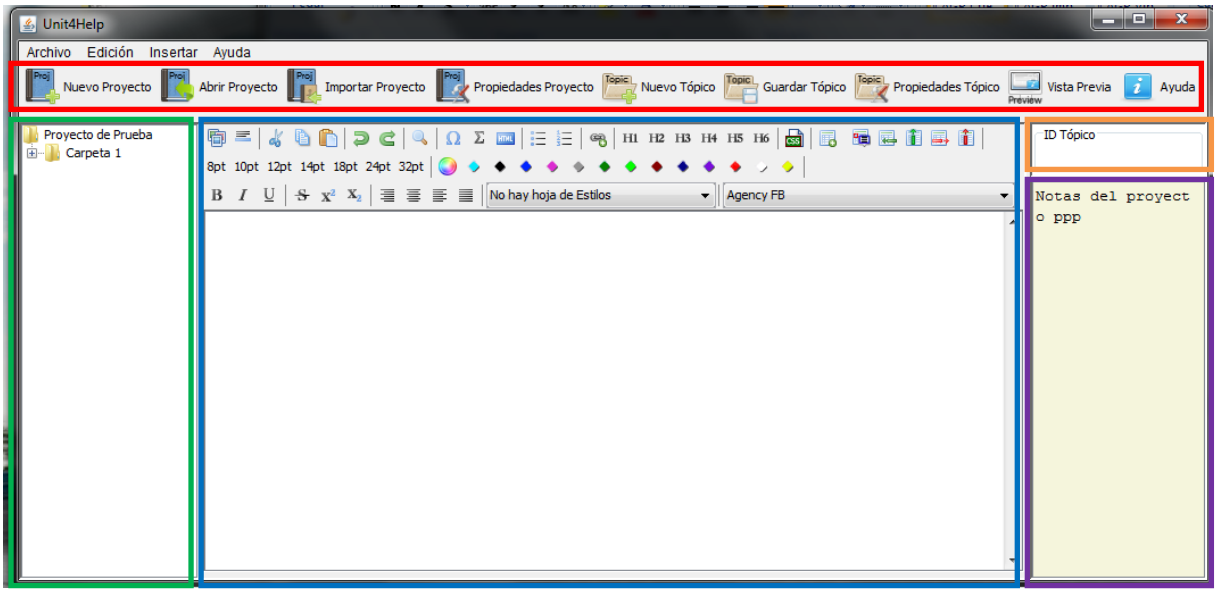

Figura 4 Interfície gràfica

Barra d'eines: S'ha decidit posar a la part superior de l'aplicació per facilitar l'accés a les funcions més habituals, i perquè els documentalistes prefereixen tenir-la a la part superior perquè els hi és més còmode i deixa més espai per l'editor. En aquest cas son les de Nou Projecte, Obrir un Projecte, Importar un Projecte, Propietats del Projecte, Nou Tòpic, Guardar Tòpic, Propietats del Tòpic, Vista Prèvia i Ajuda de l'aplicació.

Degut a que no són gaires les funcions incloses en la barra d'eines, s'ha decidit acompanyar del nom a l'icona per tal de que quedi més clar i no hagi confusió.

Arbre de continguts: Totes les aplicacions que tenien l'arbre de continguts el tenien a la part esquerra, i els documentalistes estan acostumats a tenir-ho en aquesta posició, així que és va decidir posar-ho aquí per comoditat i costums. Aquest arbre és molt semblant a un explorador d'arxius tal i com es pot veure a la figura 4 i es vol que tingui un menú emergent amb les opcions bàsiques tal com les propietats d'aquell topic, crear un nou tòpic o eliminar-ne aquest.

Editor: L'editor al ser la part principal de l'aplicació i la que més espai necessitarà, es va decidir posar al centre. Aquest editor és un component extern tal i com explicarem més endavant.

ID del tòpic: Es va decidir posar a la interfície ja que és de les poques propietats que toquen del tòpic. És un camp de text, petit i simple, ja que l'únic que ha de dur es un número enter o el nom d'un arxiu HTML que el relacionarà amb un formulari concret de l'aplicació per tal de poder anar directament de l'ajuda a l'aplicació i al revés de manera ràpida.

Notes del projecte: Es va posar en sota el ID del tòpic per ser el lloc que quedava dintre la interfície, ja que no és una part important per la creació de l'ajuda, encara que pot ser molt útil pels documentalistes.

La interfície està dissenyada amb una estructura de 3 columnes com es pot apreciar, aquestes poden ser redimensionades per l'usuari en qualsevol moment facilitant la feina que estigui duent a terme en aquell moment, depenent si està centrant l'atenció en una tasca o un altre, aquest mida es guarda en un fitxer de configuració de l'aplicació que serà carregada un cop la torni a obrir tenint les mateixes dimensions que l'ultima vegada.

Un altre punt de la interfície són les finestres de càrrega, molt útils en els processos de importació de projectes o quan s'obre un projecte, ja que poden trigar un temps d'uns minuts depenent de la mida, amb el que tenir un indicador d'aquesta carrega és d'ajuda per l'usuari.

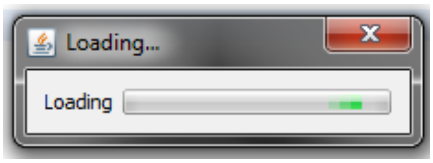

Figura 5 Finestra de càrrega

Una part molt important són les finestres de diàleg de l'aplicació per tal de fer els processos que necessitem, a continuació farem una breu explicació dels diferents diàlegs. Totes aquestes finestres han sigut creades amb l'ajuda de la llibreria JavaForms.

Importar Projecte: El diàleg de importació de projecte tenim els camps necessaris per configurar tot el projecte nou que crearem a partir d'un en format antic. Com es pot observar s'ha intentat fer una interfície molt neta i clara i en molts dels casos es completen automàticament molts camps amb l'elecció d'un sol paràmetre, faciliten i augmentant el rendiment, com per exemple un cop s'ha escollit el projecte font, automàticament s'omplen el nom del projecte, sent el mateix que l'antic, i la resta de camps s'omplen tenint en compte el espai de treball (workspace) que tenen definit en l'aplicació per tal d'importar-ho en la mateixa ruta on tenen els altres projectes.

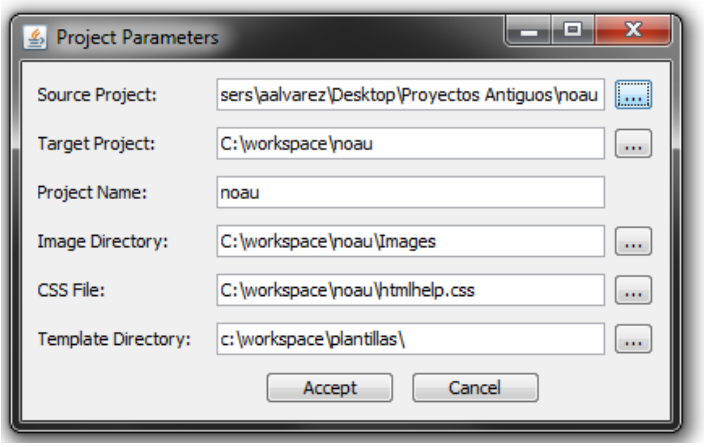

Figura Finestra de Importar Projecte

Propietats del projecte: Aquest diàleg mostra les propietats del projecte obert actualment. Segueix la mateixa estructura que tots els diàlegs, clars i amb els elements necessaris. Des d'aquí també es poden modificar les propietats com el nom, la codificació de las pàgines HTML i les rutes dels elements del projecte.

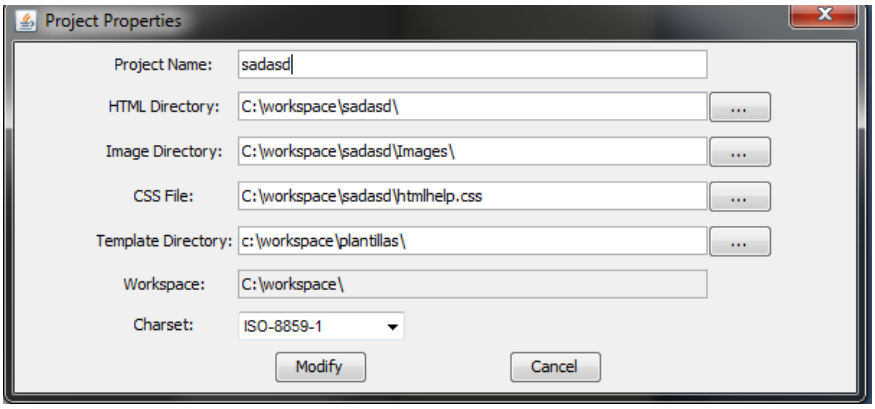

Figura 6 Finestra Propietats de projecte

Nou Tòpic: Aquesta finestra es mostra a l'hora de crear un nou topic. Es fan comprovacions de nom i de fitxer per veure que no es repeteixen, i en cas de ser així s'avisa amb un missatge per pantalla i es mostra en el requadre de la dreta els noms del arxius que coincideixen amb l' introduït. Això facilita molt la feina a l'usuari pel fet que no ha de anar al directori a mirar els arxius, i augmenta la rapidesa del procés.

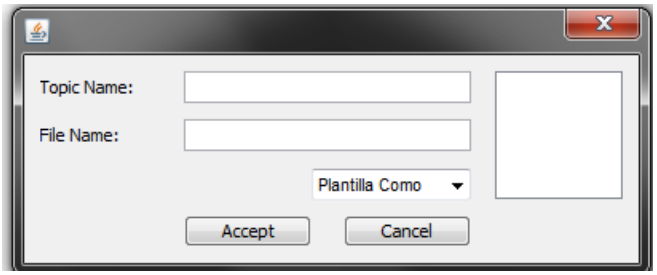

Figura 7 Finestra Nou Tòpic

Propietats del Tòpic: En aquesta finestra es pot veure i modificar el nom del tòpic i el nom del arxiu físic HTML. També mostra el ID del tòpic però aquest no és modificable des d'aquí.

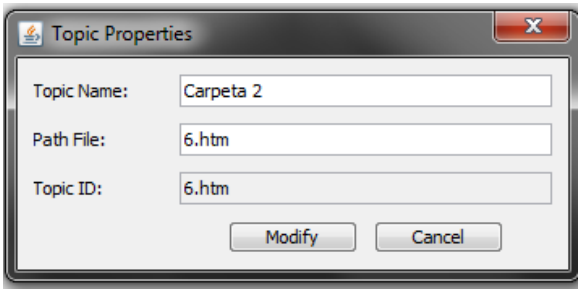

Figura 8 Finestra Propietats Tòpic

Gestor de plantilles: La finestra per gestionar les plantilles dels fitxers HTML a la hora de crear nous tòpics. Permet modificar el noms, afegir i eliminar plantilles de forma fàcil i ràpida.

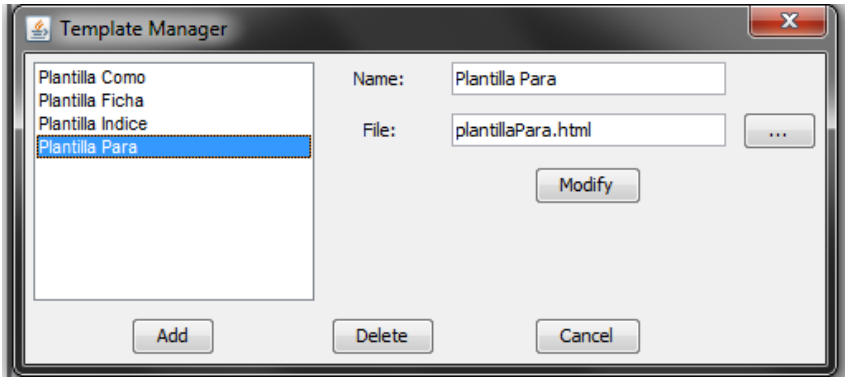

Figura 9 Gestor de plantilles

LLOC DE TREBALL (WORKSPACE)

ESTRUCTURA

En aquest projecte una de les particularitats que té, és que els documentalistes tenen un número elevat de projectes d'ajuda, així que es va decidir crear una estructura de dades dins de l'aplicació per tal de tenir en tot moment controlat el lloc on estan tots els projectes, plantilles, fulles d'estil, etc.

Aquesta estructura no deixa de ser una carpeta en un lloc determinat del disc dur triat pel documentalista. La utilitat de tenir definit un lloc de treball o workspace, és que l'aplicació mitjançant l'arxiu de configuració sap en tot moment quin és aquest

lloc de treball facilitant la feina dels documentalistes. Per exemple a l'hora de crear un projecte, l'aplicació ja sap en quina carpeta anirà i on s'ha d'ubicar, només li cal saber el nom del projecte (la ubicació del projecte en cas de que el documentalista ho necessiti pot estar fora del workspace, però per defecte sempre serà ubicada allà). Aquest workspace pot ser una unitat de xarxa o local, no hi ha cap tipus d'inconvenient en això.

LLIBRERIES EXTERNES

JDOM

Definició

JDOM és una llibreria de codi obert per a la manipulació de fitxers en format XML optimitzat per la plataforma Java. Aquesta llibreria és una modificació de la llibreria DOM del consorci World Wide Web (W3C) que es va crear per la manipulació del XML en pàgines HTML amb JavaScript, encara que es pot utilitzar la llibreria DOM en Java, té algunes mancances molt importants que JDOM supleix com per exemple que accepta noms amb espais en blancs. Aquesta és una característica molt important pel nostre projecte degut a que la majoria dels noms del tòpics estan formats per més d'una paraula, amb el que contenen espais en blancs, i en els arxius XML s'han de poder manipular aquests noms sense que doni cap tipus d'error.

També hi ha altres llibreries per manipular aquests arxius com SAX, però al igual que DOM no estan pensades per utilitzar amb cap llenguatge en concret amb el que fa que siguin més difícils de manipular envers la llibreria JDOM, així que es va decidir utilitzar aquesta.

Estructura

L'API està formada per 5 paquets, només comentarem les característiques més rellevants dels que farem servir.

Del paquet org.jdom destaquem les classes "Document" què és la que representarà el document XML virtual. La classe "Element" que representa l'element o etiqueta que formen el document XML i la classe 'Attribute' que representa els atributs que poden tenir aquestes etiquetes o elements.

Del paquet org.jdom.Adapters és la que a través del patró de disseny "Adapter" ja que JDOM no té un "parser" propi, que és l'eina que es fa servir per recórrer els fitxers, ja que JDOM té diferents parsers, i no tots tenen les mateixes funcions, així les recopila totes en classes adaptadores. En el nostre cas utilitzarem el parser "Xerces".

Després hi ha dos paquets, el org.jdom.input i el org.jdom.output, que són els que utilitzarem per crear i donar sortida a la nostra classe "Document".

Utilització

En la nostra aplicació utilitzem molts cops la llibreria JDOM, cada cop que s'ha de modificar un arxiu XML o agafar-ne informació d'un, es fa servir. La utilització és sempre igual i segueix aquesta estructura:

```
SAXBuilder builder = new SAXBuilder(false);
org.jdom.Document doc;
doc = builder.build("./conf.xml");
```

En la primera línea el que es fa és declarar el parser que farem servir per tal de reconèixer el document XML. En la segon línea creem l'objecte Document que hem esmentat abans i que serà el que farem servir per la manipulació. I la tercera línea a través del parser obté l'estructura del document que li passem al constructor ("build").

Llavors un cop tenim el document en memòria la manera de procedir es obtenir l'arrel o la etiqueta pare del XML i anar recorrent i obtenint els fills d'aquestes etiquetes per tal de manipular la informació que desitgem, com mostrem en el següent exemple.

Arxiu XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<conf>
      <proj1 text="sadasd" target="C:\workspace\sadasd\" />
      <proj2 text="MARTES" target="C:\workspace\MARTES\" />
      <proj3 text="MIERCOLES" target="C:\workspace\MIERCOLES\" />
      <workspace text="C:\workspace\" />
      <plantillas text="c:\workspace\plantillas\" />
</conf>
```

Codi:

```
Element raiz = doc.getRootElement();
proj1Link = raiz.getChild("proj1").getAttributeValue("target");
proj2Link = raiz.getChild("proj2").getAttributeValue("target");
proj3Link = raiz.getChild("proj3").getAttributeValue("target");
```

A la primera línea el que fem es agafar l'element pare del document i emmagatzemar-ho en l'element "raiz". Aleshores en les següents línees el que fem es accedir al fill de l'arrel amb el nom "proj1" en el cas de la segona línea, i demanem que ens doni el valor del atribut "target" i aquest el guardem en una variable.

Ara veurem com es fa per modificar els atributs de les etiquetes i com guardar-los posteriorment en el arxiu físic, perquè un error molt comú és modificar en memòria el contingut del document, però oblidar-se de passar-ho al arxiu físic. Farem servir el mateix arxiu XML que en l'exemple anterior.

```
raiz.getChild("proj2").setAttribute("target", targetProj1);
raiz.getChild("proj3").setAttribute("text", textProj2);
raiz.getChild("proj1").setAttribute("target", targetProj2);

XMLOutputter out = new XMLOutputter();
FileOutputStream file = new FileOutputStream("./conf.xml");
out.output(doc,file);
```

Com podem observar en les primeres tres línees el que fem es agafar les etiquetes "proj2", "proj3" i "proj1" i afegir al atribut "target" o "text" en el cas del "proj3", els texts que contenen les variables "targetProj1", "textProj2" i "targetProj2" .

En les tres últimes línees el que fem es definir la sortida del tipus XML, seleccionar el fitxer de sortida passant com a paràmetre la ruta del arxiu al "FileOutputStream" i per últim el que fem és traspassar la informació del arxiu virtual "doc" al arxiu físic "file" que hem definit.

Amb tot això que ens aporta la llibreria JDOM podem manipular de forma fàcil i senzilla els documents XML podent manipular molta informació amb poques línees i poca complexitat.

ORACLE JAVA HELP

Introducció

La llibreria d'Oracle Java Help (OHJ) és un conjunt de components Java i de diferents APIS[3] pel desenvolupament i la visualització de les ajudes en entorn Java a partir de documents HTML. OHJ està dissenyat principalment per mostrar les ajudes en les aplicacions Java.

Com hem comentat en anteriors capítols les ajudes en format Java estan formades per documents HTML que són las pàgines que es mostren al usuari final i uns arxius

[3] Una **interfície de programació d'aplicacions** o **API** és el conjunto de funcions i procediments (o mètodes, en la programació orientada a objectes) que ofereix una biblioteca per a ser utilitzada per un altre software amb una capa d'abstracció.

XML que contenen la informació de l'estructura interna. Una de les avantatges d'aquest visor és que suporta força característiques del HTML com les següents:

- Versió HTML 4.0
- Fulla d'estils CSS1 i CSS2
- Applets de Java[4]
- Sincronització amb la taula de continguts
- Vinculació dels topics amb l'aplicació a través de ID's

Totes aquestes característiques fan que el visor i les ajudes siguin molt dinàmiques per l'usuari i molt navegables, ja que no únicament ofereixen la opció de posar text pla, sinó que pots posar altres components que ajudin a l'usuari a entendre l'aplicació.

Funcionament

Per fer funcionar l'ajuda i crear el visor amb els components, l'arxiu que fa servir OHJ és el "ohs.xml" que conté la següent informació:

```
<?xml version="1.0" encoding="UTF-8" ?>
<helpset version="1.0">
<title>paau</title>
<maps>
<homeID>paau</homeID>
<mapref location="Map.xml"/>
</maps>
<view>
<type>oracle.help.navigator.tocNavigator.TOCNavigator</type><data
engine="oracle.help.engine.XMLTOCEngine">toc.xml</data>
</view>
</helpset>
```

Com podem observar aquí es defineixen on està el arxiu "Map.xml" que conté la informació dels tòpics, així com s'especifica al visor on trobar el arxiu que conté l'estructura de l'ajuda que és el "toc.xml" i que defineix el arbre de continguts de l'ajuda.

[4] Un **Applet Java** és un component d'una aplicación que s'executa en el context d'un altre programa, per exemple un navegador web.

Aquesta llibreria permet també crear índex de cerca per tal de poder trobar parts concretes dins de l'ajuda, encara que en el nostre cas no es necessari fer aquesta part ja que la nostra ajuda anirà integrada dins d'una aplicació i no se'ns demana aquesta funcionalitat.

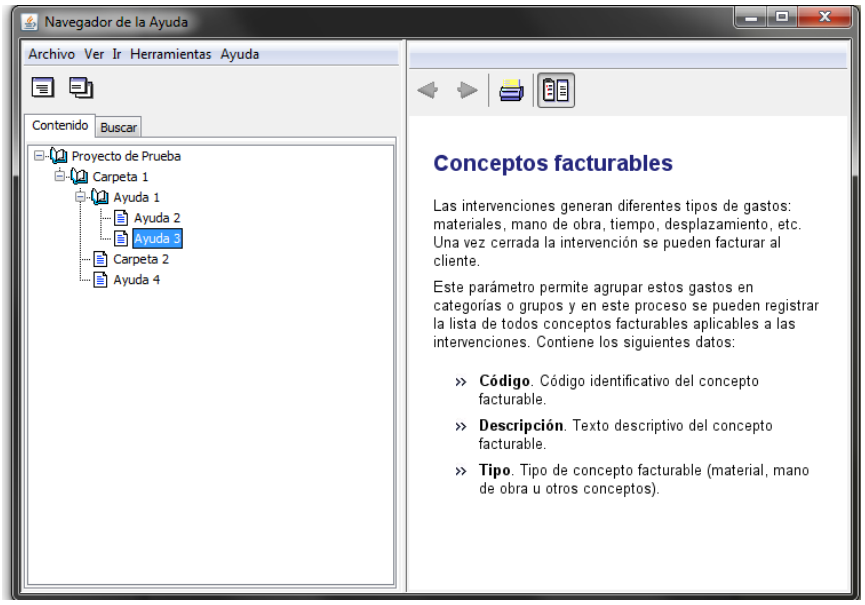

Figura 10 Visor d'ajuda

En aquesta imatge podem observar com seria l'aparença final del visor d'ajuda amb un projecte Java. A la part esquerra es pot observar el arbre de continguts amb l'estructura i els tòpics corresponents. I a la part dreta es mostra el contingut en format HTML del tòpic seleccionat.

JAVAFORMS

Introducció

JavaForms és una llibreria de lliure distribució creada pel grup JGoodies, que és dedica a la creació de components per millorar l'aspecte de les aplicacions Java i la seva usabilitat. Alguns d'aquests components no son de lliure distribució però en el nostre cas sí.

Aquesta llibreria ens permet manipular els components dintre de la nostre interfície gràfica de forma fàcil i senzilla. Un dels problemes de Java és quadrar tots els

components i alinear-los correctament, ja que hi ha diferents tipus de "FormsLayouts" que pots accedir des del "LayoutManager" que t'ajuden a col·locar els diferents components, però que son una mica complicats sobretot quan el número de components del panell és força elevat o quan els components ocupen espais molts diferents o son de diferents grandàries.

Utilització

La manera de treballar amb aquesta llibreria és semblant a treballar amb una fulla de càlcul de Microsoft Excel. El primer que has de fer és definir la quadricula que tindrà la nostra finestra i que ens servirà per després posar els components dintre d'aquestes cel·les, com es pot veure a continuació:

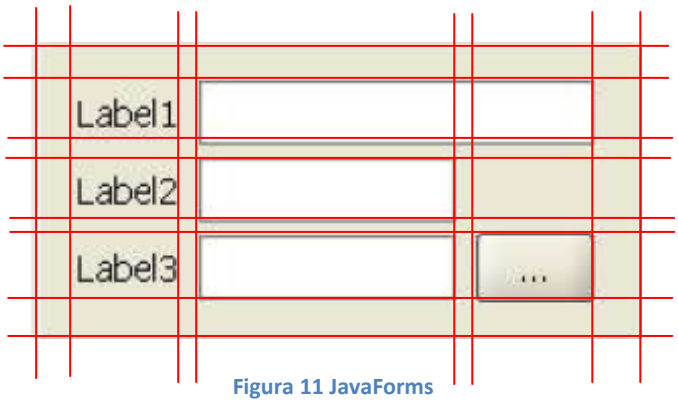

Figura 11 JavaForms

```
FormLayout layout = new FormLayout(
 "f:20px, f:75px, f:10px, f:90px, f:10px, f:50px, f:20px", //columnes
 "f:15px, f:25px, f:10px, f:25px, f:10px, f:25px, f:15px"); //files
```

Amb aquesta definició el que fem es crear la quadrícula virtual del component, ara faltaria posar els components dintre de la nostra finestra, per fer això primer hem de crear la "CellConstraints" que són les constants que ens ajudaran a definir l'estructura de la finestra, ja que com s'observa en la imatge anterior els diferents components ocupen espais i mides diferents, i assignar el "layout" que hem definit anteriorment al nostre panell o finestra. Per fer això necessitarem el següent codi:

```
JPanel panel = new JPanel(layout);

CellConstraints cc = new CellConstraints();
panel.add(new JLabel("Label1"), cc.xy (2, 2));
panel.add(textField1, cc.xyw(4, 2, 3));
panel.add(new JLabel("Label2"), cc.xy (2, 4));
panel.add(textField2, cc.xy (4, 4));
panel.add(new JLabel("Label3"), cc.xy (2, 6));
panel.add(textField3, cc.xy (4, 6));
panel.add(Button, cc.xy (6, 6));
```

Amb aquest codi creem la finestra de la figura 12. Com podem veure és molt fàcil manipular els components. Al panell o finestra afegim amb el mètode "add(component,constants)". Les constants veiem que poden tenir diversos modificadors com x, y, w i h (encara que aquesta última no surt en l'exemple). X i Y defineixen la columna i la fila on es començarà a col·locar aquest component, en el primer cas del "Label1" es començarà a posar en la fila 2 columna 2.

El paràmetres W i H de les constants fan referència al número de files (H) o columnes (W) que ocuparà el component. Per exemple tenim que el primer camp de text ocupa 3 columnes, per això a l'hora d'afegir-lo se li passa com a paràmetres "cc.xyw(4,2,3)" que ens indica que el component començarà a la fila 2 columna 2, i ocuparà 3 cel·les en direcció horitzontal tal i com es pot observar en la figura 12.

NATIVE SWING – DJ PROJECT

Introducció

DJ PROJECT és un projecte de llicència lliure que proporciona unes eines i unes llibreries per ajudar al desenvolupament d'aplicacions Java. Concretament, DJ PROJECT – NATIVE SWING és un projecte que pretén facilitar la integració de components natius dintre de les aplicacions SWING[5].

En el nostre cas ens ha servit per poder integrar un editor HTML a la nostra aplicació, ja que hem hagut d'integrar un editor escrit en JavaScript, i nativament Java no pot contenir codi JavaScript en les aplicacions Swing. Al inici vam integrar l'editor HTML EkitCore, però en la fases de proves va donar molts errors que comentarem en posteriors capítols. Així que degut a les dificultats de trobar un component Swing

[5] Swing és una biblioteca gràfica per Java. Inclou widgets per la interfície gràfica d'usuari tal com caixes de text, botons, desplegables i taules, entre altres.

d'un Editor HTML degut a que casi tots estan basats en tecnologia web (JavaScript, Phyton, etc..) aquesta eina ens ha proporcionat una via per tal de solucionar el nostre problema.

DJ PROJECT fent servir les llibreries de Native Swing, integra diferents editors HTML basats en JavaScript, proporcionant-nos les eines per poder-lo integrar en la nostra aplicació. Concretament dels 3 diferents editors que permet integrar: FCKEditor, TinyMCE Editor i DirtyIndicator, ens hem decantat pel FCKEditor degut a que és el més complet i que s'adapta millor a les nostres necessitats.

Per exemple, DirtyIndicator és un editor massa senzill, encara que incorpora suport per inserir imatges, cosa que TinyMCE no té suport i s'han d'inserir mitjançant codi font, cosa que volem evitar. Una de les funcions que té FCKEditor és la correcció ortogràfica, cosa que cap dels altres dos implementen i que és una eina molt útil. Un altre característica important del FCKEditor respecte als altres editor és el suport dels fulls d'estil (CSS) dintre del editor, cosa que va ser un dels requisits que és va demanar dins del projecte. Un altre detall important del editor, i que en aquest cas també comparteix amb TinyMCE és que permet enganxar text directament des del Microsoft Word respectant l'estil fet servir i sense perdre informació, i no només posant el text pla com fa l'altre editor.

Integració

Per integrar aquest editor en la nostre aplicació necessitem les dues llibreries bàsiques, DJNAtiveSwing-SWT.jar i DJNativeSwing.jar, a més del arxiu comprimit que contingui l'editor escollit, en el nostre cas el FCKEditor. Un cop adjuntat tot això al nostre "classpath"[6] s'ha de crear una classe del nostre editor que comentarem en el capítol de codificació, i introduir les següents línees de codi creant un objecte d'aquesta classe editor.

```
NativeInterface.open();
SwingUtilities.invokeLater(new Runnable() {
public void run() {
   newEditor = new HTMLEditor();
   newEditor.htmlEditor.addHTMLEditorListener(
       new HTMLEditorAdapter() {
       @Override
       public void notifyDirtyStateChanged(
              HTMLEditorDirtyStateEvent e){
            hayCambios = true;
```

[6] Directori arrel on es troben tots els arxius, llibreries i fitxers necessaris pel desenvolupament de la nostra aplicació.

```
                aplicacion.setTitle(aplicacion.getTitle() + "*");
            }
        });
    }
});
```

Com es pot observar és necessari sigui executat dins d'un nou "thread" per tal del seu correcte funcionament.

Un dels problemes que ens hem trobat amb aquest conjunt de llibreries, és que necessiten la última versió de la llibreria "swt.jar", que és la llibreria swing de java en la que es troben els components gràfics. El problema és que d'aquesta llibreria hi ha dues versions, la de 32 bits i la de 64 bits, amb el que depenent de l'estructura del processador que tingui la màquina es necessitarà una o un altre.

Per resoldre aquest problema s'ha creat un script per mirar la versió de la màquina virtual de Java que està instal·lada a la màquina i depenent de si és d'arquitectura de 32 o 64 bits, s'agafa una llibreria o un altre, ja que sinó dona un error d'execució en l'aplicació.

FASE DE CODIFICACIÓ I PROVES

ORGANITZACIÓ DEL CODI

En aquest apartat farem una descripció de com hem estructurat el projecte a nivell de classes en la codificació així com en els arxius o directoris complementaris que fem servir pel correcte funcionament de l'aplicació.

Estructura del projecte

Al inici del projecte vam fer un petit esbós de com tindria que ser l'estructura dels projecte, de les seves classes i els fitxers que necessitaríem. El problema és que degut a la inexperiència i als requeriments o millores que hem anat afegint al projecte, aquesta estructura s'ha modificat en gran mida, amb el que explicarem la estructura final, així com la finalitat d'aquesta.

En el nostre directori d'instal·lació trobarem els següents arxius i carpetes necessaris pel correcte funcionament de l'aplicació.

Unit4Help.jar: Arxiu generat a través de la compilació del codi font del programa necessari per l'execució.

Unit4Help.bat: Accés directe a l'execució del programa.

Conf.xml: Arxiu que conté tota la informació necessària perquè a l'hora d'executar l'aplicació pugui funcionar correctament. Aquí tenim un exemple:

```
 <?xml version="1.0" encoding="UTF-8" ?>
- <conf>
 <altura text="711" />
 <anchura text="1068" />
 <proj1 text="Projecto 1" target="C:\workspace\Proj1\" />
 <proj2 text="Projecto 2" target="C:\workspace\ Proj2\" />
 <proj3 text="Projecto 3" target=" C:\workspace\ Proj3\" />
 <workspace text="C:\workspace\" />
 <plantillas text="C:\workspace\plantillas\" />
 </conf>
```

Ejemplo\: Directori que conté tots els arxius i carpetes necessaris a l'hora de crear un nou projecte d'ajuda.

Plantillas\: Directori on es guarden les plantilles que posteriorment son copiades al espai de treball. Dintre d'aquest directori estan les plantilles així com el arxiu 'plantilles.xml' que és qui conte la informació per poder-les tractar dins de l'aplicació.

Lib\: Directori on es guarden totes les llibreries que s'han fet servir en el nostre projecte i que son necessàries pel correcte funcionament de l'aplicació.

Icons\: Directori que conté tots els icones que fa servir l'aplicació.

Files\: Directori que conté els arxius temporals que fa servir l'aplicació i també conté el arxiu comprimit que conté el editor HTML (CKEditor) que requereix.

Estructura de les classes

En aquest apartat tractarem com han estat estructurades les classes així com els packages [7]de l'aplicació.

En la nostra aplicació tenim dos packages, "FCKeditor" i "com.unit4.help". El primer conté tots els arxius JavaScript del editor, així com els arxius de configuració (que poden ser modificats per tal de que s'ajustin el màxim possible a les nostres necessitats). Aquest paquet ha sigut importat integrament ja que és un component extern que fem servir.

El segon paquet i més important, és on es troben totes les classes que hem creat i que podem observar les seves relacions en la figura 12. A continuació explicarem detalladament que fa cadascuna d'aquestes classes i mostrarem part interessants del nostre codi.

Logger.java: Aquesta classe és l'encarregada de gestionar la creació i l'actualització del log de l'aplicació. Per fer-ho l'únic que necessita és definir un seguit de paràmetres que serviran per definir el patró que seguirà el log. Per exemple, per tenir el patró següent que és el que utilitza la nostra aplicació:

```
INFO: 10:03:25 --> Class ManageHTML - Start fixFormat.
INFO: 10:03:26 --> Class ManageHTML - Finish fixFormat.
```

Hem de definir un layout (patró) com el següent:

```
PatternLayout defaultLayout = new PatternLayout("%p: %d{HH:mm:ss} --> %m%n");
```

Com podem observar el '%p' indica el tipus de missatge que és: error, info o warning.

[7] Un package (paquet) agrupa un conjunt de classes que tenen una mateixa finalitat.

El terme ‘%d’ ens indica la data i entre els claudàtors indiquem el patró de la data, en el nostre cas com en el arxiu s’especifica la data hem cregut convenient que en els missatges només mostrant l’hora és suficient per no tenir informació repetida i ocupar espai amb informació que no aporta res. I per finalitzar el terme ‘%m’ que ens indica el missatge que s’ha passat per paràmetre al log en cada crida. El paràmetre ‘%n’ únicament indica que s’ha de fer un salt de línea, per així tenir una estructura clara a l’hora de llegir el log de l’aplicació.

HTMLEditor.java: Aquesta classe és l’encarregada de crear el panell i l’editor que contindrà aquest panell. La principal característica és que degut a que és un component extern s’ha de configurar, i per això es fa servir un cadena de caràcters una mica particular en la que es poden definir des de els botons que aniran en la barra d’eines fins algunes de les seves característiques, a continuació veurem les que hem necessitats nosaltres per tal de configurar-lo a les nostres necessitats.

```
"FCKConfig.ToolbarCanCollapse = false;\n" +
"FCKConfig.FullPage = true ;\n"+
"FCKConfig.AutoDetectPasteFromWord = false ;\n" +
"FCKConfig.IncludeLatinEntities = false ;\n"+
"FCKConfig.EditorAreaCSS = '"+ ficheroCSS +"' ;\n" +
"FCKConfig.SkinPath = FCKConfig.BasePath + 'skins/silver/' ;\n" +
"FCKConfig.BaseHref = 'C:/workspace/sadasd/'";
```

FCKConfig.ToolCanCollapse: Això fa que la barra d’eines sempre estigui visible, ja que aquest editor té l’opció de que la barra pugui desaparèixer per tenir més espai per visualitzar el document.

FCKConfig.FullPage: Aquesta característica ens permet mostrar tot el codi font del document HTML quan el usuari ho necessita. En cas de que aquest atribut estigui a ‘false’ a l’hora de mostrar el codi font només es mostraria el que hi hagi entre les claus <body> i <\body>, cosa que no ens és útil degut a que molts cops necessitaran modificar les metadades del document que estan fora de aquestes claus, com el ‘DOCTYPE’ o els títols.

FCKConfig.AutoDetectPasteFromWord: Aquest atribut ens permet copiar directament des de Microsoft Word sense tenir que pressionar el botó de ‘enganxat especial des de Word’ així amb la drecera “CTRL+V” es pot enganxar directament des de Microsoft Word.

FCKConfig.IncludeLatinEntities: Aquest paràmetre ens permet que el editor tingui en comptes els caràcters especials com ‘ñ’,’<’,’>’, accents, etc. Per tal de que en el codi fon no posi una codificació especial sinó el caràcter com a tal. Ja que molts editor posen al codi font posen caràcters especials com ‘á’ és convertit a ‘&#225’ cosa que no interessa als documentalistes.

FCKConfig.EditorAreaCSS: Aquest paràmetre indica la ruta del fitxer d'estil que volem que el editor tingui en compte a l'hora de mostrar els arxius HTML independentment del que tingui l'arxiu definit.

FCKConfig.SkinPath: Aquest atribut ens permet canviar l'aparença del editor per tal de que s'assembli més a la nostra aplicació o segueixi un patró més definit.

FCKConfig.BaseHref: Aquest atribut ens defineix quina serà la nostra direcció (path) predefinida dins del editor a l'hora de fer els hipervincles o inserir imatges dins els documents HTML.

Un cop definit aquests atributs, és procedeix a crear el editor. Degut a que és un component extern hem tingut que crear nosaltres els mètodes de obrir i guardar els documents HTML editats. El que fem és agafar el contingut HTML del editor i guardar-ho en el fitxer corresponent o agafar-ho del fitxer i posar-ho en el editor en cas contrari. Aquí posem com exemple el mètode de obrir:

```
public void abrirArchivo(File archivo) throws IOException{

        FileReader n;
        n = new FileReader(archivo);
        Source source = new Source(n);
        source.fullSequentialParse();
        OutputDocument htmlDoc = new OutputDocument(source);
        String htmlFinal = htmlDoc.toString();
        htmlEditor.setHTMLContent(htmlFinal);
}
```

HTML2XML.java: Aquesta classe ens dona la funcionalitat de passar documents en format HTML a documents en format XML, respectant la jerarquia de les etiquetes. Això ens és molt útil, i l'utilitzem només en aquest cas, quan s'ha d'importar un projecte en format antic. Això és degut a que la estructura jeràrquica de l'ajuda en el format antic està en format HTML, i en format Java està en XML.

Aquesta classe s'ajuda de la llibreria Jtidy que és un parser[8] de HTML. I fa que el podem tractar de forma fàcil i senzilla els seus components per tal de poder fer la transformació de l'estructura, ja que és molt més senzill tractar un document XML que no un document HTML.

ManageHTML.java: Aquesta classe s'ha creat per tal de poder gestionar els documents HTML. En el nostre cas el únic tractament que se li fa és el de la codificació. Ja que els usuaris en qualsevol moment pot canviar la codificació dels documents. Per fer això el que fem és recórrer el directori del projecte i agafar tots

[8] Analitzador sintàctic

els documents HTML i després amb el parser Jericho, que està especialitzat en el tractament de documents HTML, busquem la etiqueta “content” que és la que conté la informació de la codificació (charset) i la modifiquem afegint la informació del charset corresponent, com mostrem en el següent tros de codi.

```
for (net.htmlparser.jericho.Element elemento : elementosMeta) {
	StartTag metaTag = elemento.getStartTag();
	Attributes attributes = metaTag.getAttributes();
	@SuppressWarnings("unused")
	Attribute content = attributes.get("content");

	Attributes atributosMeta =
source.getNextStartTag(0,HTMLElementName.META).getAttributes();
	Map<String,String> attributesMap = outputDocument.replace(atributosMeta,true);
	attributesMap.put("content","text/html; charset= " + charset);
	String ficheroSalida = outputDocument.toString();
}
```

ManageMenu.java: Aquesta classe ha sigut creada per gestionar les accions que es poden fer des del menú. És una classe molt senzilla que el únic que fa és saber quan l’usuari fa un clic un cert element del menú i llavors cridar la funció a la classe principal MainWindow que explicarem més endavant. Una particularitat, és que depenent quina sigui la opció es pregunta a l’usuari si desitja guardar el document ja que es perdrà aquesta informació, això ho fem amb el codi següent:

```
	int n = JOptionPane.showOptionDialog(frame,
		"¿ Do you want to save the current document? ?",
		"Save the current document",
		JOptionPane.YES_NO_OPTION,
		JOptionPane.QUESTION_MESSAGE,
		null,    //don't use a custom Icon
		options,  //the titles of buttons
		string1); //the title of the default button

	if (n == JOptionPane.YES_OPTION) {
		// SI VOL GUARDAR
	}
```

Un altre gestió que fa aquesta classe és que si l’usuari desitja sortir de l’aplicació es guarden en el fitxer “conf.xml” les variables de la mida de l’aplicació per així tenir-les el pròxim cop que s’executi, i també es modifica els projectes recents, d’aquests projectes en parlarem a la classe de StartWindow.

ManageToolBar.java: Aquest classe és molt semblant a la classe ManageMenu ja que en si fa el mateix però gestiona la barra d'eines de l'aplicació, que en certa manera podem dir que és un menú però només amb les opcions més comuns, així que el codi és casi el mateix. Realment és casi una duplicació de codi, i ens havíem plantejat alguna opció o estratègia per tal de no fer codi redundant. El problema és que al ser components separats i de diferent tipus és impossible gestionar a la vegada els seus esdeveniments. Així que és va decidir que la millor manera era replicar aquest codi de gestió que crida a les funcions.

ManageTree.java: Aquesta classe fa la gestió del arbre del projecte d'ajuda que mostrem a la figura següent:

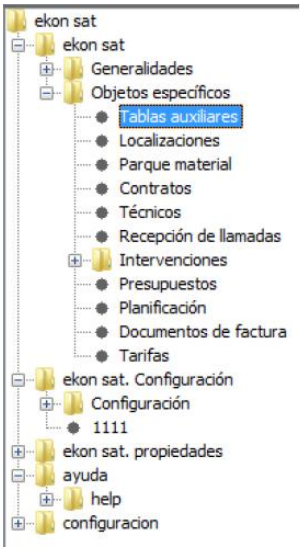

Figura 12 - Arbre de l'ajuda

A aquesta classe la cridem al inici de l'aplicació i el primer que fa és crear l'estructura d'aquest arbre a través del arxiu XML toc.xml que conté l'estructura jeràrquica de l'ajuda. Però la particularitat està en que els nodes d'aquest arbres no són noms com en els arbres normals, sinó que són objectes "tocitems" dels quals parlarem en la seva classe corresponent, ara només ens interessa saber que aquests elements guarden el nom del tòpic, el target[9] (o ID del tòpic) i la seva direcció física o URL. Per omplir l'arbre fem servir un algorisme recursiu.

Un dels problemes que vam observar va ser que quan el projecte tenia una certa grandària, al obrir els documents depenent d'on estiguessin ubicats en el arbre es trigava molt en obrir pel fet que les cerques en un arbre s'han de fer de forma recursiva i és una operació que necessita molts recursos, per això es va optar en crear

[9] El target és un identificador que fan servir les ajudes en java per poder vincular la pàgina de l'ajuda amb un punt de l'aplicació concret, per així poder anar de l'aplicació al punt de l'ajuda concret.

una estructura alternativa que pogués relacionar de forma ràpida el nom del tòpic i la seva direcció física. És va optar per la estructura HashMap de Java, aquesta estructura és un conjunt de duples <clau,valor>. Així que la única manera de accedir a un valor és mitjançant la seva clau, així podem accedir a les adreces dels fitxers mitjançant el seu nom i target de manera immediata sense tenir que fer cap cerca. El inconvenient d'això és que quan es carrega un projecte nou s'inverteix un temps en crear aquesta estructura, però que és un preu petit amb el augment de rendiment que suposa per l'usuari a l'hora de fer servir l'aplicació.

Aquesta classe també gestiona els esdeveniments que té aquest arbre com per exemple si fem doble clic sobre un tòpic s'obre en el editor. També gestiona el menú emergent que surt quan és fa clic amb el botó dret i que dona les opcions més habituals en un tòpic com són afegir un tòpic fill, eliminar el tòpic (aquest només és pot fer servir si el tòpic és un node arrel o terminal) i veure les seves propietats.

Tocitem.java: Aquesta classe ens serveix com a estructura per tal de guardar tota la informació relativa a un node o tòpic. En el nostre cas un dels problemes és que la informació relativa als tòpics es troba en diferents arxius i codificada de diferent manera, això vol dir que la opció de anar a buscar la informació dels tòpics als arxius cada cop que l'usuari la necessiti no és la forma més eficient de cara al rendiment de l'aplicació. Per això es va buscar aquesta solució, crear una estructura, un objecte que contingués tota aquesta informació, en aquest cas conté el nom del tòpic, la direcció física del arxiu HTML, i el target del tòpic. Llavors quan es carrega un nou projecte o es va creant un de nou pas a pas, el que es fa es crear objectes 'tocitems' que recopilen la informació dels diferents arxius i fa que tinguem en tot moment tota la informació referent als tòpics carregada en memòria i puguem accedir a ella ràpidament.

Tota aquesta estructura ens és molt útil sobretot a l'hora d'obrir els tòpics en el editor ja que tenim de forma molt ràpida la seva direcció física, així com a l'hora de guardar ja que s'ha de guardar tota aquesta informació en els diferents fitxers per tal de mantenir l'estructura del projecte correctament.

ProjectParameters.java: Aquesta classe fa una mica la funció d'un panell de control, ens manté en tot moment la informació relativa de l'aplicació referent al projecte obert actualment. És una manera elegant de tenir variables globals dins del projecte, ja que en Java estan prohibides com a tals.

D'aquesta classe només es crea una instancia degut a que només pot haver un projecte obert a la vegada i per tant només ens interessen els paràmetres d'aquest

projecte, quan s'obre un altre o es crea un de nou, aquests paràmetres son actualitzats en el mateix objecte.

Els paràmetres que guardem són els següents:

- Nom del projecte: Ens dona el nom identificatiu del projecte.
- Path HTML: Ens indica on estan ubicats físicament els arxius HTML.
- Path IMG: Ens indica on estan ubicades les imatges del projecte.
- Path CSS: Ens indica on està ubicat el arxiu d'estils pels fitxers HTML.
- Path Plantilles: Ens indica on estan ubicades les plantilles que poden fer servir els usuaris per crear nous tòpics.
- Notes: Ens guarda les notes del projecte en memòria.
- Workspace: Ens indica el path del espai de treball.
- Document obert: ens indica quin és el tòpic obert en el editor en aquest instant.

L'avantatge d'això és que en qualsevol punt del projecte podem accedir a aquestes variables, això fa que ràpidament puguem obtenir la ruta dels documents necessaris per tal de actualitzar l'aplicació sense tenir que fer cap accés a arxius en el disc millorant considerablement la rapidesa de l'aplicació.

WorkspaceWindow.java: Com hem explicat anteriorment la nostra aplicació ha de tenir un espai de treball assignat en un directori físic, llavors poden haver-hi dos motius pels quals no estigui definit, una instal·lació nova o que s'hagi modificat el fitxer de configuració. Si es dona un d'aquests casos l'aplicació al iniciar ho detecta i crida a aquesta classe que el que fa és demanar a l'usuari una nova definició del espai de treball i modifica el fitxer de configuració de la nova manera.

En el punt on l'usuari selecciona la ruta del workspace és molt important controlar que sempre la ruta sigui correcta ja que sinó l'aplicació donaria errors greus i el funcionament no seria correcte. Per tal de controlar això, s'obliga al usuari mitjançant un JfileChooser, figura 13, ja que amb això activant la següent opció en el codi font:

```
if(returnVal == JFileChooser.APPROVE_OPTION){
        // CODI SELECCIÓ VÀLIDA
}else{
        // CODI SELECCIÓ INVÀLIDA
}
```

Amb això el que fem es que el propi component verifica que la ruta sigui vàlida, és a dir, que sigui un patró correcte en la ruta que hagi escollit el usuari. Un cop seleccionada aquesta ruta, l'usuari encara que la pot veure en un camp de text no la pot modificar, en cas de que ho vulgui fer ho haurà de fer a través del JfileChooser,

així no hi ha cap manera possible que la ruta sigui invàlida i que no la controlem nosaltres. En cas de que l'usuari no hagi seleccionat cap, no està activa la opció de continuar endavant.

Un cop s'ha introduït aquesta informació es modifiquen els arxius de configuració amb l'espai de treball actual.

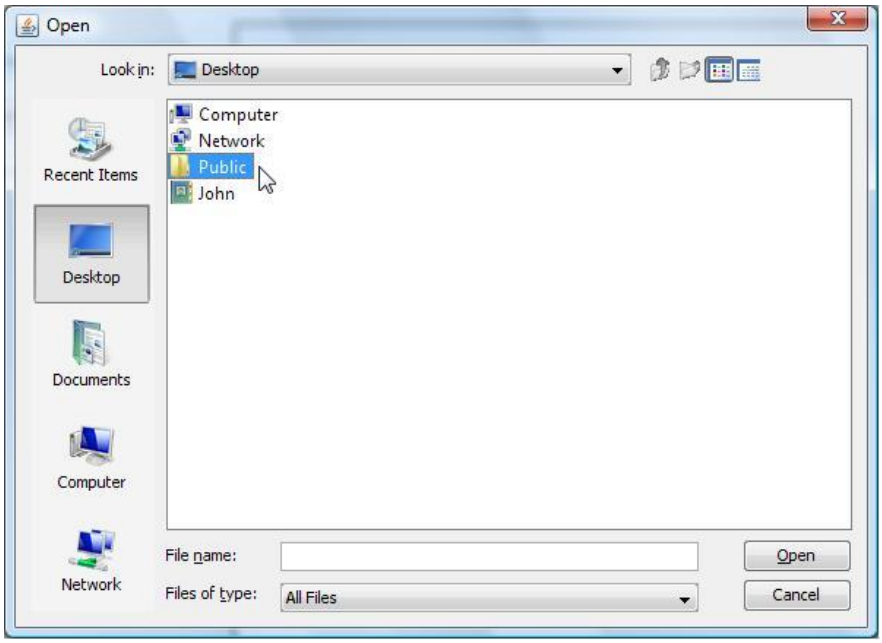

Figura 13 – JfileChooser

StartWindow.java: Aquesta classe és la que conté el 'main[10]' de la nostra aplicació. La funció principal d'aquesta classe és comprovar que existeixi un espai de treball, i verificar que el fitxer de configuració estigui correcte per tal de configurar l'aplicació.

En cas de que no existeixi el espai de treball es crida a la classe WorkspaceWindow que hem explicat anteriorment per tal de gestionar la creació del espai de treball.

Un cop hem agafat els paràmetres de configuració l'aplicació crear una finestra amb les tres opcions bàsiques que té l'usuari d'iniciar l'aplicació:

- Projectes recents: Aquí trobem els tres últims projectes que s'han utilitzat, un accés directe a ells, per poder accedir de forma rapida i automàtica.
- Des d'Arxiu: Aquí se'ns obre un explorador de fitxers i hem de seleccionar el arxiu "info.pro" del projecte que vulguem obrir.
- Projecte Nou: Des d'aquí creem un projecte nou a través d'un assistent que ens guia pels diferents passos que hem de seguir per crear un projecte nou.

[10] Inici de l'aplicació, per on comença a executar-se el codi.

Una de les funcions auxiliars que fem servir en aquesta classe, és la de copiar un directori sencer a un altre lloc del disc dur. Aquesta funció va ser creada perquè a l'hora de crear un projecte nou hem de copiar els fitxers de configuració i la carpeta d'imatge, amb el que ens és molt útil per tal de copiar-ho al espai de treball que tingui seleccionat fent els canvis corresponents en el nom del projecte.

```
public void copyDirectory(File srcDir, File dstDir) throws IOException {
        if (srcDir.isDirectory()) {
                if (!dstDir.exists()) {
                        dstDir.mkdir();
                }

                String[] children = srcDir.list();
                for (int i=0; i<children.length; i++) {
                        copyDirectory(new File(srcDir, children[i]),
                                        new File(dstDir, children[i]));
                }
        } else {
                copy(srcDir, dstDir);
        }
}
```

Com podem observar és una funció recursiva. Li entren els paràmetres del directori de origen (srcDir) i el de destí (dstDir) i el que fa és recórrer el directori de origen recursivament i copiar cadascun dels fitxers amb la funció "copy(srcDir, dstDir)".

MainWindow.java: Aquesta és la classe principal del nostre projecte, és la que s'encarrega de gestionar tota l'aplicació i és la que conté les funcions principals.

En MainWindow és on es crea la interfície gràfica i tota la seva gestió, és a dir, des d'aquí es criden a totes les classes explicades anteriorment per crear els seus respectius objectes (el editor, l'arbre de continguts, el log, etc.). També és on estan totes les funcions que es poden utilitzar des del menú o des de la barra d'eines.

No explicarem tot el codi, degut a que són més de 4000 línees, així que destacarem algunes funcions o estructures de dades que ens han sigut útils durant el projecte per fer més fàcil i optimitzar el seu rendiment.

Un de les estructura de dades que més hem utilitzat han sigut les llistes d'elements. Això ha sigut degut a que en Java aquestes llistes tenen un iterador ja implementat que et permet recórrer la llista de forma molt fàcil amb poques línees de codi. La

estructura de llista en el nostre cas la fem servir molt per tractar els fitxers XML degut a que alguns la seva estructura és lineal i per tant es pot tractar com una llista.

```
doc = builder.build(paramProyecto.getPathHTML() + "Map.xml");
Element raiz = doc.getRootElement();

/* Modificamos los elementos del xml */
List listaMap = raiz.getChildren("mapID");
Iterator it = listaMap.iterator();
boolean eliminado = false;
while(eliminado == false && it.hasNext()){
        Element nodoAux = (Element) it.next();

/* si es el nodo que buscamos */
        if(nodoAux.getAttributeValue("target").equals(targetNodo) &&
        nodoAux.getAttributeValue("url").equals(urlNodo)){

                nodoAux.setName("borrame");
                raiz.removeChild("borrame");
                eliminado = true;
        }
}
```

Com podem observar en aquest exemple, a través del primer element d'un document XML, creem una llista del seus fills. Aleshores és tan senzill com anar recorrent del primer al últim amb la instrucció "it.next()", i anar tractant cada element per separat en cada iteració del bucle. El únic inconvenient és que només pots anar en una direcció i no pots tornar enrere en la llista. En el nostre cas això no és important, perquè normalment el recorregut dels fitxers es seqüencial fins trobar el element que podem modificar amb el que ens és molt eficient.

Un altre funció interessant és la funció de reemplaçar una cadena de text per un altre dins d'una cadena. Això ens és útil quan els usuaris canvien el nom del projecte per exemple i s'han de modificar les rutes dels documents HTML, imatges i arxius del projecte. Així amb es busca el nom antic dins de la cadena de caràcters de la ruta i es modifica pel nou nom.

```
static String reemplazar(String str, String pattern, String replace) {
        int s = 0;
        int e = 0;
        StringBuffer result = new StringBuffer();

        while ((e = str.indexOf(pattern, s)) >= 0) {
                result.append(str.substring(s, e));
                result.append(replace);
                s = e+pattern.length();
        }
        result.append(str.substring(s));
        return result.toString();
}
```

Com podem veure al mètode li arriben la cadena “str” que conté la cadena original, la cadena “pattern” que conté el nom original que volem modificar i la cadena “replace” que conté la nova cadena que volem introduir. Això també ens és útil perquè quan hi ha canvis en un formulari en temps real, es pugui modificar la informació dels altres camps de text per fer més eficient la feina del usuari i evitar errors en les modificacions.

Un altre del temes que hem verificat en totes les classes però més especialment en aquesta degut a la seva dimensió i funcions que fa, és la gestió d’errors “try/catch[11]” per tal de controlar l’aplicació quan hi ha algun error no controlat. Mostrem un exemple a continuació:

```
try {

        file = new FileOutputStream(pathPlantillas + "plantillas.xml");
        out.output(doc,file);

} catch (Exception e1) {

        e1.printStackTrace();

        JOptionPane.showMessageDialog(null, e1.getMessage() + " Error in Manager
        Template. Please check the template path in 'info.pro' file " +
        "and check 'plantillas.xml' file information.", "Unit4Help Error",
        JOptionPane.ERROR_MESSAGE);

        System.exit(-1);
}
```

Com podem observar en el bloc “try” que seria on aniria el codi que volem executar s’intenta crear un fitxer i assignar-li un contingut. Si aquest codi falla i llença una excepció per exemple, si no a trobat el fitxer perquè ha sigut eliminat accidentalment o perquè hi ha hagut un desbordament del buffer de sortida, aleshores s’executaria el bloc “catch”.

En aquest cas imprimeix el “stacktrace” que seria la llista de classes per on ha passat l’execució fins arribar al punt que ha generat l’excepció. Aleshores s’avisa al usuari del error que ha succeït mitjançant una finestra emergent amb un missatge d’error i en aquest cas concret com és un error greu i es recomana reiniciar l’aplicació es surt de l’aplicació.

[11] Mètode per tractar les excepcions que es poguin donar durant l’execució del codi i pogui donar a un error.

Un altre tema important que gestiona aquesta classe és la gestió dels canvis en els documents per tal de saber si s'ha de guardar o no els canvis. El mètode fàcil seria sempre que sortim de l'aplicació o es canviï de document, guardar, independentment de si s'han fet canvis o no, el problema és que això no és òptim degut a que el procés de guardar un document és dels que més pes tenen en l'aplicació degut a que han de recórrer i modificar diferents fitxers, algun d'ells de maner recursiva, amb el que és una carrega important pel sistema. Llavors es va decidir comprovar en aquests casos quan es necessari guardar el document: quan s'ha modificat el contingut del editor i quan s'ha modificat el ID del formulari. Quan es canvien les propietats del document no és necessari degut a que en aquell moment ja son modificades les propietats en els documents per tal de poder seguir oferint la vista prèvia en qualsevol instant.

Per tal de fer això el nostre editor incorpora un "listener[12]" que detecta quan hi ha hagut un canvi en el editor i llença l'esdeveniment corresponent i fa que s'executi un codi concret, com veiem a continuació.

```
newEditor.htmlEditor.addHTMLEditorListener(new HTMLEditorAdapter() {
        @Override
        public void notifyDirtyStateChanged(HTMLEditorDirtyStateEvent e){
                hayCambios = true;
                aplicacion.setTitle(aplicacion.getTitle() + "*");
        }
});
```

Com podem veure creem al nostre editor un "listener" perquè escolti al component per si hi ha alguna notificació. Aleshores, el codi que va dins no s'executa fins que el editor llença aquest esdeveniment del "listener" i s'executa tants cops durant l'execució com faci és llenci aquest esdeveniment. En aquest cas nosaltres notifiquem al usuari que hi ha dades que guardar posant un "*" al títol de l'aplicació que mostra per defecte el document obert en aquell moment. A part de posar una variable de control "hayCambios" amb el valor "true" per tal de poder fer un control més exhaustiu.

Per últim destacar que el mètode d'importar projecte ha tingut que ser creat dins d'una classe interna per motius de rendiment i es cridat mitjançant una nova tasca o "thread"[13].

[12] Mètode o conjunt de mètodes que incoporen les classes de Java que notifiquen o envien un missatge intern conforme s'ha produït un esdeveniment, 'escolten' aquest component per si envia algún missatge.

[13] Fil d'execució d'una aplicación, poden haver-hi diferents threads en execució dins d'una mateixa aplicación.

A continuació per acabar veurem un gràfic de les classes i de les seves relacions. Com es pot observar hi ha moltes classes que no estan relaciones amb cap classe ni amb la classe principal, degut a que no són objectes que fem servir durant l'execució, sinó que fan funcionalitats concretes i no necessiten estar relacionades.

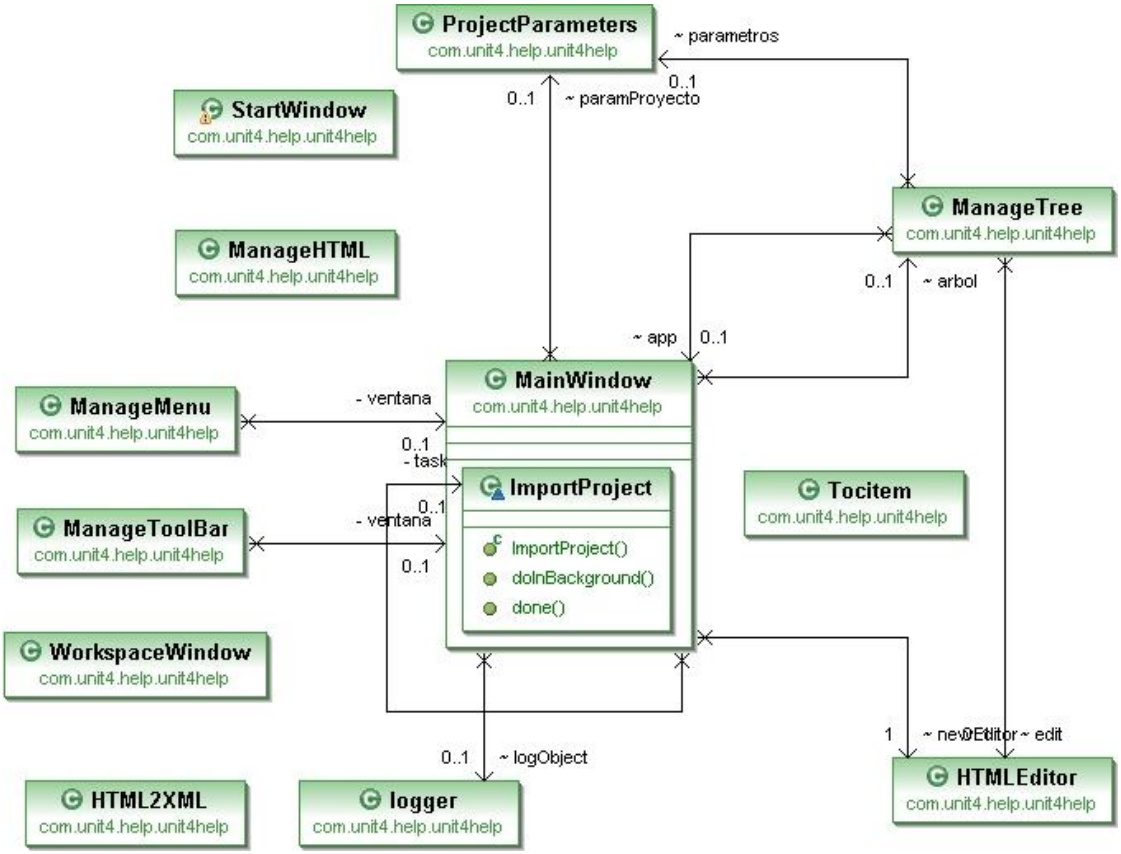

Figura 14 - Diagrama de classes

ESTIL DE CODIFICACIÓ

Durant la codificació del projecte no s'ha seguit cap patró en concret, ja que dins de l'empresa no exigeixen la utilització de cap, ni fan servir cap en concret. Encara que s'ha intentat seguir un patró per tal de que el codi sigui fàcil de llegir i de mantenir tant durant la creació del projecte com en futures ampliacions o modificacions.

Per exemple coses que s'han anat respectant al llarg de la codificació han sigut els noms de las variables, mètodes i classes. Les variables sempre comencen en

minúscula i en cas de que siguin dos paraules las següents comencen en majúscula, per exemple “hayCambios” que hem vist anteriorment. Els mètodes segueixen el patró de les variables en el tema de minúscules i majúscules i en ambdós casos s’intenta que el nom sigui identificatiu de la funció que té aquesta variable o mètode per tal de que sigui més fàcil la lectura.

Les classes en canvi totes les paraules comencen amb majúscules com per exemple “ManageHTML” també s’ha intentat que el nom reflecteixi al màxim possible la funció de la classe per tal de que el projecte pugui ser fàcil de realitzar i buscar coses concretes.

Sempre que s’ha entrat en una funció o en un bucle, el codi que va dins ha sigut indentat amb un tabulador. Com podem observar a continuació.

```
public void copyDirectory(File srcDir, File dstDir) throws IOException {
	if (srcDir.isDirectory()) {
		if (!dstDir.exists()) {
			dstDir.mkdir();
		}

		String[] children = srcDir.list();
		for (int i=0; i<children.length; i++) {
			copyDirectory(new File(srcDir, children[i]),
					new File(dstDir, children[i]));
		}
	} else {
		copy(srcDir, dstDir);
	}
}
```

També una cosa que podem observar del codi anterior, es quan s’obre un claudator “{“ per una funció o un bucle, sempre es posa en la mateixa línea que la definició del bucle o classe.

També una de les cosses que s’ha tingut en compte es comentar molt el codi, a part de JavaDoc que comentarem més endavant, dins de les funcions o en algunes variables que tenen alguna funció una mica especial i que només amb el nom no s’acaben de definir correctament, s’ha fet a la línea superior un comentari d’aquest estil:

```
/* Comentari per definir la linea inferior */
String lineaInferiorDeCodi;
```

També s'ha comentat algun tros de codi que fa alguna funció una mica especial o difícil de veure només amb el codi per tal de si algun dia s'ha de revisar, ampliar o modificar s'entengui bé per a qualsevol programador.

JAVADOC

Javadoc és una utilitat de Oracle per a la generació de la documentació de API's en format HTML. Javadoc és el estàndard per a la documentació de les classes de Java, casi tots els editors ja els generen automàticament.

Per generar aquests documents HTML, es fan servir unes etiquetes especials en el codi font, que estan precedides pel caràcter '@'. Aquestes etiquetes s'han d'escriure al inici de cada classe, mètode o variable que vulguem incorporar en el nostre javadoc, és recomanable que es comentin totes les classes i mètodes per tal de fer més fàcil la seva entesa en futures revisions. Aquestes etiquetes han d'estar introduïes en els comentaris que comencen per "/**". En la següent taula explicarem les etiquetes que es poen posar i que nosaltres hem fet servir en el nostre projecte:

Tag	Descripción
@author	Nombre del desarrollador.
@deprecated	Indica que el método o clase es antigua y que no se recomienda su uso porque posiblemente desaparecerá en versiones posteriores.
@param	Definición de un parámetro de un método, es requerido para todos los parámetros del método.
@return	Informa de lo que devuelve el método, no se puede usar en constructores o métodos "void".
@see	Asocia con otro método o clase.
@throws	Excepción lanzada por el método
@version	Versión del método o clase.

Figura 15 - Javadoc

PROVES

En aquest apartat veure'm el tipus de proves que s'han dut a terme i en que ens han servit per tal de millorar l'aplicació.

Proves unitàries

Durant tota la fase de codificació del projecte s'han anat fent proves unitàries sobre els mètodes i funcionalitats conforme s'anaven creant i codificant, per tal de comprovar el correcte funcionament de cada part del codi. El disseny d'aquestes proves ha sigut molt variat depenent de la funcionalitat de cada mètode.

Això ens ha servit per trobar errors dintre de l'aplicació i per poder fer les restriccions correctament, sobretot en els casos en que l'usuari ha d'introduir informació o dades mitjançant els diàlegs en l'aplicació i que ha de tenir un format concret o aportar una informació d'una manera molt exacta.

Un exemple clar d'això es a l'hora d'introduir la ruta el que volem que sigui el nostre espai de treball, perquè tot funcioni perfectament la ruta ha de tenir el següent patró:

" [lletra_unitat]:\ ruta_completa\directori_de_treball\ "

Si no és així, l'aplicació generaria problemes a l'hora de trobar les plantilles, el fitxer de codificacions, a l'hora de crear nous tòpics, entre altres coses, amb el que és molt important fer unes bones proves unitàries per verificar que no puguin introduir cap dada que estigui fora d'aquest patró.

Encara que en aquest projecte no s'han fet gaire proves unitàries com a tals, ja que la majoria de mètodes van lligats entre sí i sobretot necessiten la informació inclosa per l'usuari. Degut a això ens hem centrat sobretot en controlar de manera efectiva la informació que introdueix el usuari, perquè així ens assegurem que si aquesta informació segueix els patrons que ha de seguir, l'aplicació no fallarà. Igualment, s'han fet proves perquè el resultat sigui el desitjat per l'usuari, encara que com veurem en el següent apartat nosaltres no tenim la perspectiva que té un usuari, així que vam decidir que el departament de documentació, és a dir, l'usuari final, provés l'aplicació en diferents fases del procés, per tal de comprovar la funcionalitat i els errors de funcionalitat que puguin haver.

Proves del departament de documentació

El departament de documentació, que seran els usuaris finals de l'aplicació han sigut una peça molt important durant les proves. Durant el cicle de desenvolupament de l'aplicació, és van anar traient diferents versions beta[14], concretament durant tot el projecte s'han tret 6 versions, cadascuna afegint funcionalitats i corregint els errors de les versions anterior.

De cada versió portàvem un control de les funcionalitats noves que s'havien afegit i dels errors corregits. Per exemple de la versió 0.1 a la versió 0.2, que van ser les dues primeres, van haver-hi aquestes:

- Creació d'un log de l'aplicació
- Funcionalitat "Obrir Projectes"
- Funcionalitat "Crear Projectes"
- Funcionalitat "Crear Tòpics"
- Modificació dels Id's dels formularis
- Funcionalitat de "Vista Prèvia"

En aquesta versió per exemple ja teníem uns errors coneguts del editor que vam començar a utilitzar d'inici (Editor Ekitcore) i gràcies a les proves que el departament va fer vam detectar molts errors en el editor, que jo com a desenvolupador no vaig saber veure. Molts eren errors a l'hora de crear llistes enumerades, i amb unes propietats de les imatges molt particulars. Cosa que va fer que ens reviséssim el codi del editor que és un component extern i vam veure que no estava suficientment clar i documentat com per poder-ho solucionar i ens va fer que tinguéssim que buscar un altre que s'adaptés a les exigències del departament.

També ens han ajudat molt aquestes proves les a fer que l'aplicació sigui més fàcil per l'usuari, degut a que les seves proves a l'hora de fer servir les funcionalitats de l'aplicació ens feien arribar suggerències per tal de millorar el seu rendiment. Per exemple, a l'hora de crear un tòpic els usuaris han d'escollir una de les plantilles que tinguin instal·lades a l'aplicació perquè sigui la base del nou tòpic. Al inici això ho fèiem mitjançant un checkbox amb el nom de la plantilla com es mostra a la següent figura:

[14] Versió inacabada d'una aplicació però que té unes funcionalitats bàsiques.

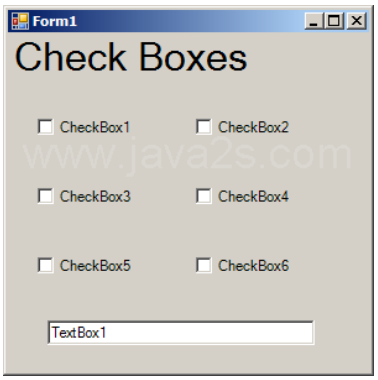

Figura 16 – Checkbox

Això els resultava incòmode a part de que si en un futur es volia ampliar això no seria eficient ja que s'hauria de crear la interfície dinàmicament dependent de les plantilles. Així que van proposar que fos una llista desplegable per tal de ser més còmode, i a més es va desenvolupar un gestor de plantilles tal i com s'ha explicat en la part de disseny i codificació.

I com aquest petits detalls sobre el disseny de l'aplicació que s'han anat modificant durant el transcurs del seu desenvolupament per tal de millorar l'experiència de l'usuari.

Proves de rendiment

Proves de rendiment com a tals només hem fet una que és a l'hora d'importar projectes que és el procés més laboriós que es duu a terme. Per tal de fer-les es va demanar al departament el projecte més gran que tenien actualment en ús en la format antic. Aquest projecte estava composat per més de 1500 pàgines en HTML, i els seus respectius arxius de configuració tal i com s'ha explicat al inici de la memòria.

En total el temps d'exportació no va arribar al minut, i després d'analitzar tot el procés amb el departament i el nostre ajudant tècnic vam arribar a la conclusió de que era un temps més que acceptable i vàlid. També es van fer proves amb projectes de menys embergadura i el temps en tots els cassos va ser menor.

Aprofitant l'importació d'aquest projecte de gran embergadura es van fer proves a l'hora d'obrir aquest projecte i de realitzar modificacions en els arxius més al fons del arbre de continguts que en teoria és el que més temps triga en fer un recorregut de manera recursiva. En tots els casos els temps van ser més que acceptables, sobretot després del canvi que vam fer amb els 'hashmap' tal i com hem comentat a l'apartat de codificació, que feia que el rendiment fos més elevat.

CONCLUSIONS

En aquest apartat farem una explicació de les conclusions que hem pogut treure al llarg del desenvolupament del projecte tant a nivell tècnic de planificació com a nivell personal i fer un petit resum de com s'ha acabat el projecte.

OBJECTIUS ASSOLITS

L'objectiu principal que era el de desenvolupar una aplicació pel departament de documentació que unifiqués tot el seu procés d'ajuda i fes el pas a la nova tecnologia Java que utilitza l'empresa ha sigut assolit. També podem dir que ha sigut assolit amb casi totes les exigències que requerien així com també ampliacions i millores que han anat sorgint al llarg del projecte.

Ara els usuaris del departament de documentació, tenen en un sola aplicació la possibilitat de crear, modificar, ampliar i visualitzar els projectes d'ajuda de l'empresa. A més, poden de manera fàcil importar els projectes en format antic al nou amb la mateixa aplicació sense que hagin de fer cap pas entremig.

Amb el que el principal objectiu que era unificar el procés de treball i augmentar el rendiment i la comoditat dels treballadors del departament de documentació ha sigut assolit.

AMPLIACIONS

Ampliacions no es poden fer gaires o no estan previstes que es facin degut a que s'han afegit totes les funcionalitats que els documentalistes han suggerit per tal de desenvolupar la seva tasca amb el model d'ajuda que fan servir avui dia a Unit4.

Sí que es podrien fer algunes millores, en certs components de l'aplicació, com per exemple dotar de més dinamisme al arbre de contingut, ja que no s'ha pogut fer que puguin moure blocs de topics des d'un punt de l'estructura del arbre a un altre.

Tret d'això les ampliacions o millores jo crec que depenen més de si l'empresa fa canvis en l'estructura de les ajudes, ja que sinó, l'actual aplicació desenvolupada compleix amb totes les funcions necessàries.

PLANIFICACIÓ DEL PROJECTE

La planificació del projecte va ser força encertada i s'ha seguit en gran part. Encara que hem après molt dels errors i sobretot de les variabilitats que poden haver. Al inici per la falta d'experiència no tens en compte coses o variables que et poden fer retardar en el projecte i que veus que al llarg del projecte et penalitzen. Sobretot errors d'enteniment entre les parts que defineixen el projecte o petits detalls sobre la tecnologia Java que fa que després hagis de modificar gran part del codi degut a un error en la falta de planificació o error durant la fase de disseny que no s''ha tingut en compte.

Encara que el més important de la planificació del projecte és l'experiència que aporta cometre aquests errors i tenir el suport i l'experiència del tutor tècnic fa que aprenguis molt sobre com portar la planificació i com pots solucionar els problemes que van sorgint.

VALORACIÓ PERSONAL

A nivell personal he aprés molt en totes les fases de disseny de software. Hi ha molts aspectes que a la universitat no aprens i que només aprens un cop estàs dins del mon laboral, i tenir la possibilitat de fer un projecte des de cero passant i planificant totes les fases del projecte et fa trobar-te en moltes situacions que normalment a la universitat no et trobes, i et fa adquirir una experiència i uns coneixements molt valuosos. També la interacció amb persones que ja tenen un cert nivell d'experiència en el món laboral i concretament en el teu camp et fa veure com treballen, com pensen i com intenten resoldre els problemes que van sorgint i et fa aprendre a veure les coses o encarar els problemes de manera diferent o intentant tenir una visió més amplia i objectiva. Amb el que resumint estic molt satisfet del treball fet i la experiència adquirida durant el transcurs del projecte.

AGRAÏMENTS

Per finalitzar la memòria voldria donar les gràcies a tothom qui m'ha ajudat, suportat i donat ànims durant tots aquests anys...

A tota la meva família començant per la meva mare Juani i el meu germà Aitor que m'han suportat durant tot aquest trajecte.

Al meu pare pels consells que només un pare pot donar.

Als meus avis Tomàs i Poli, perquè sempre han estat quan els he necessitat i que segurament sense ells no hauria acabat la carrera.

Al meu tiet 'Tete', per donar-me alguna cleca quan m'ho he guanyat.

Al Toni per donar-me suport i tranquil·litat.

Al Xavi, que no és família, però com si ho fos, per aconsellar-me i ajudar-me en tants moments dolents. I als seus amics i ara també meus Jordi, Joan i Raúl per haver-me aconsellat en algunes decisions.

Als meus amics de sempre per donar-me ànims i suportar-me que no és fàcil, i per escoltar els problemes del projecte encara que no s'enteressin de res.. Miguel, Samantha, Carlos, Tamara, Lidia i Xavi.

Als meus tutors Jordi Pons, Oscar Lechago, Lluís Gómez i sobretot Josep Miquel Garcia per ajudar-me quan estava encallat i no trobava la manera de sortir del pou.

Gràcies a tots.